UNISYS

CTOS®

Batch Manager II

Installation, Configuration, and Programming Guide

Release 2.3

Priced Item

September 1992

Printed in U S America
4393 1633–000

UNISYS

CTOS®

Batch Manager II

Installation, Configuration, and Programming Guide

Release 2.3 September 1992

Printed in U S America
Priced Item 4393 1633–000

The names, places and/or events used in this publication are not intended to correspond to any individual, group, or association existing, living or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any indivdual, living or otherwise, or that of any group or association is purely coincidental and unintended.

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Comments or suggestions regarding this document should be submitted on a User Communication Form (UCF) with the CLASS specified as "Documentation", the Type specified as "Trouble Report", and the product specified as the title and part number of the manual (for example, 4393 1633-000).

UNISYS

Product Information Announcement

○New Release ● Revision ○ Update ○ Errata

Title:

CTOS® Batch Manager II 2.3 Installation, Configuration, and Programming Guide

This Product Information Announcement (PIA) announces the release and availability of the *CTOS Batch Manager II Installation, Configuration, and Programming Guide,* release 2.3, part number 4393 1633-000.

This Guide is a revision of the 2.2 release of the *CTOS Batch Manager II Installation, Configuration, and Programming Guide.*

This Guide documents the new features in the 2.3 release of CTOS Batch Manager II. It also contains corrections to the 2.2 release of the *CTOS Batch Manager II Installation, Configuration, and Programming Guide.*

To order additional copies of this document:

- United States customers, call Unisys Direct at 1-800-448-1424.
- All other customers, contact your Unisys Sales Office.
- Unisys personnel, use the Electronic Literature Ordering (ELO) system.

Please address all technical and documentation comments relative to this release to:

Unisys Corporation
Product Information
5155 Camino Ruiz
Camarillo, CA 93012

Software Release Announcements Only:	All Announcements:	System:	CTOS®
BTO56	BTO55	Release:	Batch Manager II 2.3 (September 1992)
		Part Number:	4393 1633-000

Page Status

Page	Issue
iii	Original
iv	Blank
v through xiii	Original
xiv	Blank
xv	Original
xvi	Blank
1-1 through 1-5	Original
1-6	Blank
2-1 through 2-5	Original
2-6	Blank
3-1 through 3-6	Original
4-1 through 4-15	Original
4-16	Blank
5-1 through 5-7	Original
5-8	Blank
6-1 through 6-84	Original
A-1 through A-11	Original
A-12	Blank
B-1 through B-9	Original
B-10	Blank
C-1 through C-5	Original
C-6	Blank
Glossary-1 through 12	Original
Index-1 through 6	Original

About This Guide

This guide contains descriptive and procedural information for using Batch Manager II, the batch processing utility for workstations and shared resource processors (SRPs) using the CTOS operating system. It includes software installation instructions, rules for creating Job Control Language (JCL) files, and procedures for processing batch jobs in both foreground and background partitions.

Who Should Use This Guide

The information in this guide addresses the needs of programmers and system administrators who create, use, and maintain batch processing files.

If you are unfamiliar with the CTOS operating system, you may find it helpful to review your CTOS operating system documentation before you use this guide.

How to Use This Guide

If you are using Batch Manager for the first time, you should read Section 1. It contains an overview of Batch Manager functions and features.

To find definitions of terms used in this guide or related to this product, refer to the glossary. To locate specific information, use the index.

How This Guide Is Arranged

This guide is divided into sections, with related subjects grouped together. This guide is arranged as follows:

- Section 1 provides an overview of Batch Manager operations and describes the basic concepts of the product.
- Section 2 outlines procedures for installing and removing Batch Manager software.
- Section 3 describes the system initialization and system services after you install Batch Manager.
- Section 4 outlines procedures for using Batch Manager commands to activate, display, and control batch job operations.
- Section 5 describes batch operations in the foreground (primary) partition and background (secondary) partitions.
- Section 6 explains JCL syntax, system constants and variables, and statement usage.
- Appendix A provides troubleshooting information and lists status messages Batch Manager can display.
- Appendix B describes the batch queues required for system operation.
- Appendix C provides system initialization and JCL examples.

This guide also contains a glossary of terms and an index.

Conventions

The following conventions are used:

- Preceding a JCL statement with a dollar sign ($) is optional. In the text, statements do not have the dollar sign; however, in the examples, the dollar sign is included for all JCL examples.
- Material enclosed in braces, { }, indicates that the programmer must make a choice from the options within the braces.
- Material enclosed in square brackets, [ ], indicates that the material is an option the programmer can include or omit.
- The term server (formerly master) refers to the hub of a cluster configuration.
- The term client refers to a workstation in a cluster configuration.
- The term service refers to a program or subprogram that performs and manages tasks for other programs.
- The XE530 is referred to as a shared resource processor (SRP).

Related Product Information

Table 1 lists related products and where you can find information about them.

Table 1. Related Product Information

Product	Source
CTOS I 3.4, CTOS II 3.4, CTOS III 1.0, and CTOS/XE II 3.4 operating systems	CTOS System Administration Guide
	CTOS Executive User's Guide
	CTOS Executive Reference Manual
	CTOS Operating System Concepts Manual
CTOS Editor	CTOS Editor User's Guide
BNet II	CTOS II BNet II Installation, Administration, and Configuration Guide
CTOS Queue Manager	CTOS Operating System Concepts Manual
Event Manager	CTOS Event Manager Programming Reference Manual
XE shared resource processor	XE500 Installation and Implementation Guide
CTOS Access Control	Aministration Guide

Contents

Section 4. Using Batch Manager Commands

Section 5. Batch Processing in the Foreground and Background Partitions

Section 6. Job Control Language Statements

Figures

Tables

Section 1
Overview

Batch Manager is the CTOS batch-language processor and job manager. You can use Batch Manager to:

- execute a sequence of Executive commands at a specified time
- execute a series of your own programs at a specified time
- install system services automatically on a server or workstation at boot time

Batch Manager uses statements in Job Control Language (JCL) files to execute other applications.

Batch Manager operates on all hardware supported by the following operating systems:

- CTOS/XE II 3.4 (or greater)
- CTOS I 3.4 (or greater)
- CTOS II 3.4 (or greater)
- CTOS III 1.0 (or greater)

Batch Manager can execute in the foreground partition, where you use the keyboard and display, or background partitions where the application's keyboard inputs are embedded as bytestreams in the JCL file. The SysIn facility reads the bytestream and the SysOut facility directs the video output bytestreams to a disk file or the following default file:

[Scr]<Batch>JCLFileName.sysout$$Date-Time.log

Batch Log File

Each active background Batch Manager maintains a Batch Log file (for example, **[Scr]<Batch>Batch00.log**). This is a record of all the jobs processed in the order in which they were processed. The Batch Log file includes:

- the time Batch Manager was installed
- each job's start and finish times
- the termination code for the last step of each job
- any Log statement messages in the JCL files

Time/Date Function

Batch Manager maintains a Time/Date function that allows you to schedule execution of a background batch job at any date and time (for example, Thu Jan 2, 1992 4:40 pm or 1/02/92 16:40).

Batch Manager Commands

You use the following Batch Manager commands to activate, display, and control batch operations through the Executive:

- The **Batch Foreground** command initiates a job in the foreground partition.
- The **Install BS Batch** command allows Batch Manager to run programs in background; the command installs SysIn and SysOut bytestreams for batch processing in background partitions. This is required on protected mode operating systems; it is unnecessary on real mode operating systems.
- The **Install Batch** command creates a background partition and installs a Batch Manager in it.
- The **Batch** command queues a job for background execution.
- The **Batch Status** command displays the status and scheduling details of jobs in a background partition, permits the removal of partitions and cancellation of jobs, and allows the printing of batch log files.

Job Control Language Files

Job Control Language (JCL) files are text files created by one of the following methods:

- using the Editor
- using a CTOS word processor (such as OFIS Document Designer)
- using the Command File Editor to output command forms to a JCL file

Batch Manager reads the statements within a JCL file to direct the various processing operations under its control, such as establishing the job and the user name, loading and activating the run files, and passing run-time parameters to the appropriate batch job steps.

Batch Manager processes JCL files sequentially. However, you can optionally specify control statements to alter normal processing flow and provide conditional looping and conditional execution.

Batch Jobs and Job Steps

A batch job is a sequence of JCL files processed under the control of Batch Manager. A sequence, in this instance, refers to a single JCL file, or one JCL file calling other JCL files before permanently exiting Batch Manager and returning to the Executive.

Each JCL file instructs Batch Manager to execute specific job steps. Each job step is the execution of a statement in the JCL file.

For example, you can use a batch job to compile source code, and then link the object code to produce an executable run file. The compilation is the first job step, the linking is the second.

Batch Processing in the Foreground Partition

When Batch Manager runs in the foreground partition, it processes only one batch job before returning control to the previous exit run file. A job consists of the statements beginning with the **$job** statement and continuing through the **$end** statement. Batch Manager ignores all **Log** statements and does not update the Batch Log file.

Batch Processing in a Background Partition

When Batch Manager runs in a background partition, it repeatedly polls the Queue Manager to obtain the next JCL file for processing. Batch Manager then records each job processed in the Batch Log file, noting job start and finish times and the termination status code for each job executed.

Memory and Disk Requirements

Batch Manager requires approximately 2000 sectors of disk space for storage.

Table 1-1 lists Batch Manager-related memory requirements:

Table 1-1. Batch Manager and Related Memory Requirements

Component	Minimum Memory Required, in Kbytes
Batch Foreground	300
Background processing:	
Batch Manager	300
Queue Manager	43
Byte Stream service (SysIn/SysOut)	10*
Batch Supervisor	25*
(supports multiple Batch Managers)	

* Fixed memory size

A foreground job may require additional memory to execute large JCL files.

Depending on your background tasks and their size in a JCL stream, you can increase memory for better performance by specifying a larger memory size for the Install Batch command.

Enhancements over Batch Manager II 2.2

This release of Batch Manager II provides the following enhancements:

- Extension of % Metacharacter

  In addition to identifying a parameter (%n, where n is numeric), the percent sign also identifies an external variable (%variable, where variable begins with an alphabetic character). Any variable that begins with a percent sign refers to an external variable.

- Addition of the **DeviceType** statement

  This statement takes a device name as an argument and returns a numeric device type.

- Addition of the **FileOpenStatus** statement

  This statement indicates whether or not a file can be opened in a particular mode.

- Access Control interoperation

  When Batch Manager detects the presence of Access Control, it registers each background batch job with Access Control. Once the job has completed, Batch Manager deregisters it. The Batch command now includes the optional parameter [User Role] for use by Access Control.

- User-configurable SysOut (SysInit.log) file for SRPs

  When using Batch Manager on an SRP, you can now specify a SysOut file other than **[Sys]<Sys>SysInit.log**. **[Sys]<Sys>SysInit.log** remains the default.

Section 2
Software Installation and Removal

This section describes the procedures for installing and removing Batch Manager software on your workstation.

You can install Batch Manager on systems using the following operating systems:

- CTOS/XE II 3.4 (or greater)
- CTOS I 3.4 (or greater)
- CTOS II 3.4 (or greater)
- CTOS III 1.0 (or greater)

The distribution diskettes are write-protected. Do not write-enable them or use them as working copies. Instead, use the **Floppy Copy** command to make copies of the distribution diskettes and store the originals in a safe place. Use the copies to install Batch Manager.

Note: *Before you begin installing or deinstalling Batch Manager software, use the Partition Status command to ensure that the background application BatchMgr.run is not running. Otherwise, you will receive the message* ***Erc 220 File in use.***

Installing Batch Manager from Installation Diskettes

This installation method uses the **Floppy Install** command to install the software from the Batch Manager installation diskettes. There is one 3-1/2 inch diskette and two 5-1/4 inch floppy diskettes.

Using the 3-1/2 Inch Diskette

To install Batch Manager from the installation diskette, use the following procedure:

1. Insert the Batch Manager software diskette into the diskette drive.
2. Type **Floppy Install** on the Executive command line and press **GO.**

   The system starts Installation Manager. Installation Manager displays windows and prompts for your responses.
3. When the system displays Installation Defaults, choose one of the following:

   - To accept the installation defaults, select the **Continue Installation** option and press **GO.**
   - To view or change the installation defaults, select the **Examine/Change Defaults** option, press **GO,** and view or change the defaults. Press **GO** when you are satisfied with the installation parameters.

   The system displays a sequence of installation statements and informs you when installation is complete.
4. Remove the Batch Manager software diskette and store it in a safe place.
5. If you have installed Batch Manager on a shared resource processor, reboot the system.

Using 5-1/4 Inch FLoppy Diskettes

To install Batch Manager from the 5-1/4 inch installation diskettes, use the following procedure:

1. Insert the first Batch Manager software diskette into the diskette drive [f0] and close the door.
2. Type **Floppy Install** on the Executive command line and press **GO.**

   The system starts Installation Manager. Installation Manager displays windows and prompts for your responses.
3. When the system displays Installation Defaults, choose one of the following:

   - To accept the installation defaults, select the **Continue Installation** option and press **GO.**
   - To view or change the installation defaults, select the **Examine/Change Defaults** option, press **GO,** and view or change the defaults. Press **GO** when you are satisfied with the installation parameters.

   The system displays a sequence of installation statements.
4. When the system displays the prompt to insert the second Batch Manager installation diskette, remove the first diskette.
5. Insert the second Batch Manager diskette into the diskette drive.

   The system continues its display of installation statements and informs you when installation is complete.
6. Remove the second Batch Manager software diskette and store both diskettes in a safe place.
7. If you have installed Batch Manager on a shared resource processor, reboot the system.

Installing Batch Manager from a Server

If Installation Manager has been used to install Batch Manager on a server, you can download the Batch Manager software to a locally booted cluster workstation.

To install Batch Manager software on your workstation, use the following procedure:

1. At the workstation, type **Server Install** on the Executive command line and press **GO.**

   The system starts Installation Manager. Installation Manager displays windows and prompts for your responses.

2. When the system displays Installation Defaults, choose one of the following:

   - To accept the installation defaults, select the **Continue Installation** option and press **GO.**
   - To view or change the installation defaults, select the **Examine/Change Defaults** option, press **GO,** and view or change the defaults. Press **GO** when you are satisfied with the installation parameters.

   The system displays all the software that was publicly installed.

3. Select the **Batch Manager** option and press **GO.**

   Installation Manager installs the software at your local workstation.

Removing Batch Manager Software

If Batch Manager has been installed using Installation Manager, you can use Installation Manager to remove Batch Manager software.

To remove Batch Manager software, use the following procedure:

1. Type **Installation Manager** at the Executive command line and press **GO.**

   The system displays the **Software Operation** menu.

2. Select the **Remove Installed Software** option and press **GO.**

   The system displays the **Remove Installed Software** menu.

3. Choose one of the following:

   - If Batch Manager was installed publicly, move the cursor to the **Public Software** option and press **GO.**
   - If Batch Manager was installed privately, move the cursor to the **Private Software** option and press **GO.**

   The system displays all the software that has been installed through Installation Manager.

4. Select the **Batch Manager** option and press **GO.**

   Installation Manager removes all Batch Manager software except for Batch.run, BatchMsg.bin, Cli.run (which is only installed on SRPs) and the **Batch Foreground** command.

Section 3
System Initialization

This section describes the process of installing system services on your workstation or XE shared resource processor using a JCL file. For further information on system initialization, you can refer to your *CTOS System Administration Guide*.

System Services for a Workstation

Following initialization, the operating system automatically chains to the run file specified as the chain file parameter in the system build configuration. The default file is **[Sys]<Sys>SysInit.Run**.

After the operating system loads but before you sign on, a system initialization file can run a batch job to start system services and applications in background partitions.

For example, you could create the following system initialization file to load the X-bus Interface, Telephone Service, Scaling Font Service, and the Mouse Service:

```
$JOB SysInit
$ContinueOnError
$RUN [Sys]<Sys>XBif.run
$RUN [Sys]<Sys>TMService.run
$RUN [Sys]<Sys>ScalingFontService.run
$RUN [Sys]<Sys>Mouse.run
$END
```

On workstations that boot locally, the initialization file must be named **[Sys]<Sys>SysInit.jcl**.

On workstations that boot from the server, the system searches for the initialization file in the following sequence:

1. [Sys]<Sys>HWnnn>SysInit.jcl (nnn is the workstation Hardware ID)
2. [Sys]<Sys>WSnnn>SysInit.jcl (nnn is the processor ID)
3. [Sys]<Sys>WS>SysInit.jcl

For example, for a workstation that boots locally, you could create the following initialization file:

[Sys]<Sys>SysInit.jcl File:

```
$JOB  SysInit
$RUN  [Sys]<Sys>InstallBatchBS.Run
$RUN  [Sys]<Sys>InstallBatch.Run
$RUN  [Sys]<Sys>Mouse.Run
$END
```

For a workstation that boots from the server, you could create initialization files as follows:

[Sys]<Sys>HW101>SysInit.jcl File:

or

[Sys]<Sys>WS240>SysInit.jcl File:

or

[Sys]<Sys>WS>SysInit.jcl File:

```
$JOB  Install
$RUN  [Sys]<Sys>InstallBatchBS.Run
$RUN  [Sys]<Sys>InstallBatch.Run
$RUN  [Sys]<Sys>Mouse.Run
$END
```

System Services for the SRP

In operating systems prior to CTOS/XE II 3.0, each processor board in an SRP had its own JCL and CNF files that it used for initialization. With the release of CTOS/XE II 3.0, these files were replaced by a single SysInit.jcl file and one SRPConfig.sys file.

There are three keyswitch positions on the front of the shared resource processor base enclosure. Each keyswitch position has a **SysInit.jcl** and **SRPConfig.sys** file associated with it contained in the <Sys> directory of the [Sys] volume. The keyswitch positions and their associated files are:

- **m** (for **m**anual)

  The files associated with this position are **SysInit.m.jcl** and **SRPConfig.m.sys**.

- **r** (for **r**emote)

  The files associated with this position are **SysInit.r.jcl** and **SRPConfig.r.sys**.

- **n** (for **n**ormal)

  The files associated with this position are **SysInit.n.jcl** and **SRPConfig.n.sys**.

The three positions allow you to boot the system with different mixes of system services so that you can diagnose and test for problems.

> **Caution:**
>
> Unisys recommends that you not edit or delete the files associated with the Remote keyswitch position, **SysInit.r.jcl** and **SRPConfig.r.sys**, so that if you experience a problem booting the system with the keyswitch in the Normal position, you can boot the system with the key in the Remote position.

If the system cannot find the files associated with the keyswitch position, it looks for a batch file named **[Sys]<Sys>SysInit.jcl**. If it finds it, it runs a Batch stream. Regardless of whether it finds an initialization file, the operating system starts the signon program.

The files accessed during the the boot sequence are, in order:

1. **[Sys]<Sys>SysInit.k.jcl** (k is the letter corresponding to the keyswitch position)

   or **[Sys]<Sys>SysInit.jcl** and,

2. **[Sys]<Sys>Signon.run**

On a shared resource processor running the CTOS/XE 3.4 operating system, the Batch Manager on the GP00 processor board controls the services and jobs on the other processor boards.

At system initialization time, Batch initially runs on the XE 530 GP00 processor board. During initialization, the Command Line Interpreter (CLI.run) verifies that each of the subordinate boards has sufficient memory available for Batch and then loads Batch.run on these boards.

You can include following JCL statements in the initialization file and specify a board ID, and the system will execute the statement on the specified processor board:

- **$Command**
- **$Run**
- **$RunNoWait**
- **$Call**
- **$CallNoWait**
- **$End**

The operating system searches for the configuration file corresponding to the keyswitch position. It it does not find it, it uses **[Sys]<Sys>SRPConfig.sys**. If neither is found, the system boots with a minimum board configuration.

The sequence is as follows:

1. **[Sys]<Sys>SRPConfig.k.sys** (k is the letter corresponding to the keyswitch position)
2. or **[Sys]<Sys>SRPConfig.sys**
3. or minimum board configuration

You can refer to Appendix C for a sample shared resource processor SysInit.jcl file.

Specifying a SysOut File

You can specify a SysOut file in the **Job** statement to obtain a log file of the initialization sequence by typing three commas after the job name, then the log file name. The default SysOut file is **[Sys]<Sys>SysInit.log**.

If the **<Sys>** directory is password-protected, you can specify a SysOut file in an unprotected directory.

For example, a SysInit.jcl file that writes to a log file named SysInit.log in a nonpassword-protected directory **<$000>** would look like this:

```
$Job SysInit,,,[Sys]<$000>SysInit.log
$Run [Sys]<Sys>InstallQmgr.Run, yes, 20
```

Also, if the **<Sys>** directory is password-protected, you can add a caret (^) to the end of the log file name, followed by the password. The password does not display on the screen or in the log file.

For example, a SysInit.jcl file that writes to a log file named **SysInit.new.log** in a password-protected **<Sys>** directory would look like this:

```
$JOB SysInit,,,[Sys]<Sys>SysInit.new.log^Password
$RUN [Sys]<Sys>InstallQmgr.Run, yes, 20
```

System Failure

If the operating system reloads through a bootstrap operation following a system failure, **[Sys]<Sys>SysInit.run** displays the system failure status.

If the chain from **[Sys]<Sys>SysInit.run** to Batch fails or if **[Sys]<Sys>Batch.run** does not exist, **[Sys]<Sys>SysInit.run** exits with an appropriate status code to **[Sys]<Sys>Signon.run** without processing the SysInit.jcl file.

Section 4
Using Batch Manager Commands

This section contains general information and procedures for using the commands to control the Batch Manager environment:

- **Batch Foreground**
- **Install BS Batch**
- **Install Batch**
- **Batch**
- **Batch Status**

Priority Parameter

Both the **Install Batch** and **Batch** commands have a priority parameter. The priority parameter of the **Install Batch** command sets the (operating system) processor priority of the partition in which Batch Manager is running. The priority parameter of the **Batch** command sets a priority to batch jobs within a batch queue, determining the order that Queue Manager queues the jobs.

Queue Manager

You must first install the Queue Manager before using background batch functions. You must install Queue Manager on the server of a cluster. It can also be installed on a stand-alone system. The Queue Manager executes jobs with the same priority on a first in first out basis.

Dynamic Queues

A Batch queue is required for background processing. Having dynamic queues relieves you of the responsibility of editing the Queue.Index file (**[Sys]<Sys>Queue.Index**) prior to installing Queue Manager. To take advantage of this, specify the number of required dynamic queues when installing Queue Manager.

When the **Install Batch** command is executed, Batch passes data to the Queue Manager so it can dynamically create a queue for background processing.

However, the Queue.Index file is still an important part of background Batch processing, especially for remote processing across a network. An example of a Queue.Index file of this type is included in Appendix B.

Batch Foreground Command

The **Batch Foreground** command activates Batch Manager, allowing you to execute jobs in the foreground partition.

*Note: The **Batch Foreground** command is the only command needed for foreground processing.*

To execute a batch job, use the following procedure:

1. Type **Batch Foreground** on the Executive command line.
2. Press **RETURN.**

   The following **Batch Foreground** command form appears:

   Batch Foreground
   JCL File
   [Parameters]
   [Restart JCL file?]

3. Enter the following information:
   - **JCL file**

     Enter the name of the JCL file to be processed.
   - **[Parameters]**

     Enter the parameters to be passed to the JCL file.
   - **[Restart JCL file?]**

     If you enter **Yes,** Batch Manager looks up the last RestartLabel executed during the previous run. The current JCL file execution begins from that point forward. For more information, you can refer to the description of **RestartLabel** in section 6.

     The default, **No,** executes the JCL file from its beginning.
4. Press **GO.**

Install BS Batch Command

The **Install BS Batch** command installs the Batch bytestream system service, which is necessary for protected mode operation. The Batch bytestream system service contains the SysIn and SysOut input and output bytestream facilities.

The SysIn bytestream facility allows keyboard bytestream input to be taken from a file, and the SysOut bytestream redirects the video bytestream outputs to a file. To install the SysIn and SysOut bytestream facilities, you must invoke the **Install BS Batch** command before invoking the **Install Batch** command.

To install the SysIn/SysOut bytestream facilities, use the following procedure from the primary partition:

1. Type **Install BS Batch** on the Executive command line.
2. Press **GO.**

Install Batch Command

You use the **Install Batch** command to create a background partition (minimum 300K) and install Batch Manager in it. You can create as many batch partitions as your system's memory allows, and you can specify the size of each partition.

When you install Batch Manager in a background partition, Batch Manager uses a scheduling queue to obtain JCL files to process. Because several Batch Managers on different workstations may serve the same scheduling queue, each Batch Manager obtaining entries from the queue must have access to all files required to execute a job.

If you are installing a Batch Manager that is dedicated to a particular function, you should specify a queue name. If not, you can accept the default.

To install Batch Manager in a new background partition, use the following procedure:

1. Type **Install Batch** on the Executive command line.
2. Press **RETURN.**

   The following **Install Batch** command form appears:

   Install Batch
   [Memory size (default = 300K)]
   [Batch Queue (default = BATCH)]
   [Partition name (default = BatchPart00)]
   [Batch manager priority (default = 129)]

3. You can enter the following parameters:

   - **[Memory size (default = 300K)]**

     This is the desired partition memory size, in kilobytes (minimum 300K); it should be larger than the largest application running in the Batch partition.

   - **[Batch queue (default = BATCH)]**

     This is the desired scheduling queue served by Batch Manager (for example, Joe). Batch uses Queue Manager to dynamically allocate its own queues.

- **[Partition name (default = BatchPart00]**

  This is the partition name assigned to the Batch processing partition. If the default partition, BatchPart00, is already assigned, the system installs Batch Manager in BatchPart01 (or if 01 is also already assigned, BatchPart02 is used, etc.). If you assign a partition name, the name must be unique.

- **[Batch manager priority (default 129)]**

  This sets the (operating system) processor priority of the partition in which the Batch Manager is running. You can enter values from 65-254; 65 being the highest priority, 254 the lowest. Execution time is allocated on a priority basis; eligible processes with higher priority run before those with lower priority.

4. Press **GO.**

Batch Command

The **Batch** command allows you to queue jobs to be processed in a background partition or to execute a foreground batch job.

To execute a batch job in a background partition, use the following procedure:

1. Type **Batch** on the Executive command line.
2. Press **RETURN.**

   The following **Batch** command form appears:

   Batch
   JCL File
   [Parameters]
   [Batch queue]
   [After date/time]
   [Priority]
   [Repeat After Time]
   [Job Expiration Date]
   [User Role (for Access Control)]

3. You can enter the following parameters:

   - **JCL file**

     Enter the name of the JCL file to be processed.

   - **[Parameters]**

     You can enter parameters to be passed to the JCL file.

   - **[Batch Queue]**

     You can enter the name of a selected scheduling queue (for example, Joe). If you do not specify a queue name, the Batch job executes in the foreground.

   - **[After date/time]**

     You can enter the earliest date and time Batch Manager can execute the job. The default is the current date and time (immediately). If you enter only a date or time, Batch uses the current system date or time in place of the omitted date or time.

   - **[Priority]**

     You can enter a queuing priority from 0 (highest) to 9 (lowest) that Queue Manager uses to runs the jobs. The default is 5.

     Priority has no effect unless jobs are submitted at the same time. You can specify the time of execution. If you do not specify a time of execution, Batch uses the current system time.

   - **[Repeat After Time]**

     This is time when the job is to be repeated in the format dd:hh:mm (days, hours, minutes). You can specify up to 65,535 minutes; that is, 45 days, 12 hours. For example:

     - 30:0:0 means repeat in 30 days
     - 0:0:06 or 6 means repeat in six minutes
     - 0:24:0 or 24:0 means repeat in 24 hours (1 day)

- **[Job Expiration Date]**

  This is the latest date/time the job can execute. The default is the current date/time. If the entry in this field is only partially complete, Batch derives the missing date or time from the system clock and adds this information to the entry.

  A job queued with a repeat after time will be removed from the queue at that time.

- **[User Role (for Access Control)]**

  This is the optional user role for a background batch job. When the job starts, Batch Manager registers the subject with the local access-control kernel; when the job is completed, Batch Manager deregisters it.

4. Press **GO.**

Batch Status Command

When you have a job running in the background partition, you use the **Batch Status** command to display the status of a job, including job details and the scheduling queue, cancel a job, remove a batch partition, or print a batch log file. This command interfaces with any Batch Manager and executes through any workstation in the cluster configuration.

Batch Scheduling Queues

To display the status of all batch scheduling queues, use the following procedure:

1. Type **Batch Status** on the Executive command line.
2. Press **GO.**

   The Main Batch Status display appears (refer to Figure 4-1).

   Note: *If only one Batch Manager is running, the Batch Status Queue is displayed (refer to the following procedure).*

The Main Batch Status display contains the following information:

- **Batch Queue**

  This is the name of each scheduling queue serviced by Batch Manager.

- **Status**

  This indicates whether the queue is idle (waiting) or executing.

- **Manager**

  This is the two-digit identifier (for example, 00, 01) of the Batch Manager servicing the queue.

- **Current Job**

  This is the name of the job (as it appears in the Job statement) being executed.

  Note: *If the message **BadJob** appears in this column, Batch has detected an error which will be written to the log file. Refer to Appendix A for information on reading the log file.*

  Use the **NEXT PAGE** key to scroll through consecutive screen displays; use **PREV PAGE** to return to the previous screen.

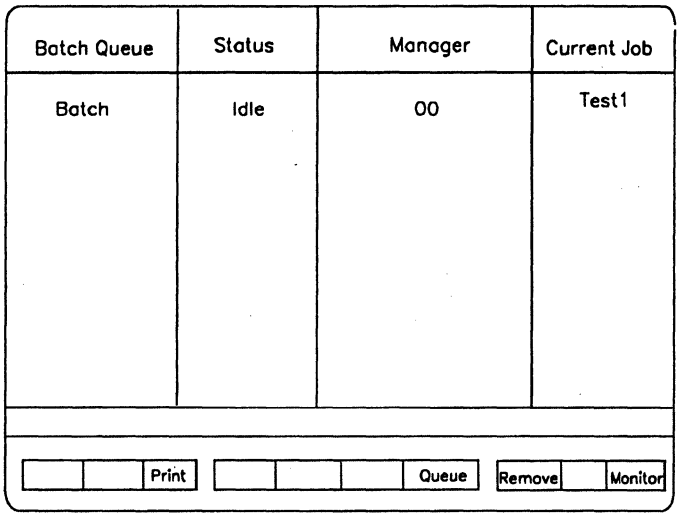

Figure 4-1. Batch Status Main Display

Displaying the Status of a Queue

To display the status of a specific batch scheduling queue, use the following procedure:

1. Complete steps 1 and 2 of the procedure for displaying the status of all batch scheduling queues.
2. On the Batch Status Main display (refer to Figure 4-1), use the UP ARROW and DOWN ARROW keys to highlight (select) the desired Batch Scheduling Queue.
3. Press the **Queue** function key (**F7**).

The display for selected Batch queue appears (refer to Figure 4-2).

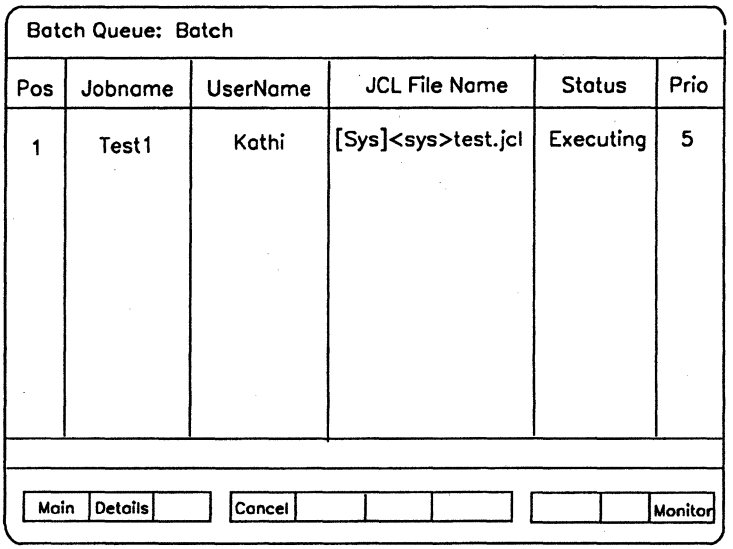

Figure 4-2. Batch Status Queue Display Screen

The following information displays:

- **Pos** (Position)

  This is the job's position in the processing sequence.

- **JobName**

  This is the name of the job, as it appears in the Job statement.

- **UserName**

  This is the name of the user as identified in the job statement.

- **JCL File Name**

  This is the name of the JCL file being processed.

Note: *You should avoid executing two jobs with identical names in the same Batch queue, since Batch will incorrectly display identical job status listings for the two jobs regardless of the actual status of the jobs.*

- **Status**

  This indicates whether the job is waiting or executing.

- **Prio** (Priority)

  This is the job's queuing priority (0 is highest, 9 lowest).

Use the **NEXT PAGE** key to scroll forward through consecutive screen displays. Use **PREV PAGE** to scroll backward through consecutive displays.

To return to the Main Batch Status display, press the **Main** function key (**F1**).

Viewing the Details of a Queue

To view details about jobs in a selected Batch status queue, use the following procedure:

1. Display the status of the selected Batch Queue (you can refer to the procedure for displaying the status of a specific batch scheduling queue, if you need help).
2. At the Batch Status Queue Display, press the **Details** function key (**F2**)

The details for the jobs in the queue appear (refer to Figure 4-3).

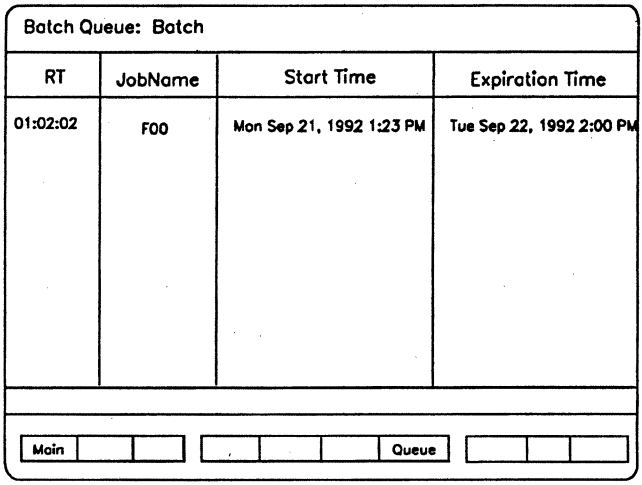

Figure 4-3. Batch Queue Details Display

The following information displays:

- **RT** (Repeat Time)

  This is the job's Repeat Time in the form: days:hours:minutes. If this field is empty, the job will execute once.

- **JobName**

  This is the name of the job as it appears in the Job statement.

- **Start Time**

  This is the start time of the job.

- **Expiration Time**

  This is the time the job will be removed from the queue. If this field is empty, the job will execute once.

Use the **NEXT PAGE** key to scroll forward through consecutive screen displays. Use **PREV PAGE** key to scroll backward through consecutive displays.

1. To return to the Batch Status Main display, press the **Main** function key (**F1**).
2. To return to the Batch Status Queue display, press **Queue** (**F7**).

Removing a Batch Manager

To remove a Batch Manager from a partition, use the following procedure:

1. Complete steps 1 and 2 of the procedure for displaying the status of all batch scheduling queues.
2. On the Main Batch Status display (refer to Figure 4-1), use the **UP ARROW** and **DOWN ARROW** keys to highlight (select) the Batch Manager to be removed.
3. Press the **Remove** function key (**F8**) to remove the selected Batch Manager upon completion of the current job step.

Monitoring the Current Job

To monitor the current job step being executed by Batch Manager, from the Main Batch Status display, use the following procedure:

1. Complete steps 1 and 2 of the procedure for displaying the status of all batch scheduling queues.
2. On the Batch Status Main display (refer to Figure 4-1), use the **UP ARROW** and **DOWN ARROW** keys to highlight (select) the Batch Manager to be monitored.
3. Press the **Monitor** function key (**F10**).

   This function displays only the following job steps:

   $Job
   $Run
   $Command
   $End

To monitor the current job step being executed by Batch Manager from the Batch Status Queue screen, use the following procedure:

1. Complete steps 1 through 3 of the procedure for displaying the status of a specific batch scheduling queue.
2. On the Batch Status Queue display (refer to Figure 4-2), use the **UP ARROW** and **DOWN ARROW** keys to highlight (select) the Batch Manager to be monitored.
3. Press the **Monitor** function key (**F10**) to display the current job step being executed by the selected Batch Manager.

Printing a Batch Log File

To print a batch log file, use the following procedure:

1. Complete steps 1 and 2 of the procedure for displaying the status of all Batch scheduling queues.
2. On the Batch Status Main display (refer to Figure 4-1), use the **UP ARROW** and **DOWN ARROW** keys to highlight (select) the Batch Manager for log file printing.

3. Press the **Print** function key (F3) to print the log file of the selected Batch Manager.

   The batch log file is directed to the print queue **[Spl]** to be printed. However, if **[Spl]** is not the default printer, the log files are stored in the **[Scr]<Batch>** directory under the default name **BatchLog$$hh:mm:ss.tmp** (you can replace Batch with another name.) Once you print a log file, you start a new Batch log.

Canceling a Batch Job

To remove or cancel a specific batch job, use the following procedure:

1. Complete steps 1 through 3 of the procedure for displaying the status of a specific batch scheduling queue.
2. On the Batch Status Queue display (refer to Figure 4-2), use the **UP ARROW** and **DOWN ARROW** keys to highlight (select) the job to be removed or canceled from the queue.
3. Press the **Cancel** function key (**F4**) to either remove a job awaiting execution or cancel the executing job after the current job step finishes.

> **Caution:**
>
> Batch Manager cancels the first matching job name entry in the queue.

For online help for any form in the Batch Status display, press the **Help** key.

Section 5
Batch Processing in the Foreground and Background Partitions

This section describes operating characteristics and functional considerations for batch processing in the foreground and background partitions. It includes a description of the batch log file.

Batch Processing in the Foreground Partition

Batch Manager is invoked in foreground using the **Batch Foreground** command, specifying the name of the JCL file to be executed, the parameters to be passed to the JCL file, and whether or not the **Restart JCL File** option should be invoked. It can also be invoked with the **Batch** command if a Batch queue is not specified.

The foreground partition allows you to interface with the program through the keyboard and video display. When the JCL file executes (refer to Example-1), the output and termination code appear on the video display (refer to Example-2); the termination code of Executive commands being executed is also directed to the SysOut file if one is specified (refer to Example-3).

Example-1: JCL File

```
$JOB  PascalTest,username,,PascalTest.SysOut
;           This job compiles, links, and runs a Pascal Program
$RUN  [Sys]<Sys>PascalFE.Run,PascalTest.Pas
$RUN  [Sys]<Sys>Linker.Run,PascalTest.Obj,PascalTest.Run
$RUN  PascalTest.Run
$END
```

Example-2: JCL File Output Screen Display

```
$JOB  PascalTest,username,,PascalTest.SysOut
Execution begun: Sat May 11, 1991 7:23 PM
          ;This job compiles, links, and runs a Pascal Program
$RUN  [Sys]<Sys>PascalFE.Run,PascalTest.Pas
5.1.0
Pass One      No Errors Detected

          Code Area Size = #0064 ( hex)
          Cons Area Size = #004A ( hex)
          Data Area Size = #0002 ( hex)

          Pass Two      No Errors Detected

          Termination status code: 0

$RUN  [Sys]<Sys>Linker.Run,PascalTest.Obj,PascalTest.Run
          Linker 8.0.1

          Termination status code: 0

          $RUN  PascalTest.Run

          Termination status code: 0

          $END

          Execution ended: Wed Sep 23, 1992 7:25 PM
```

Example-3: SysOut File Output

```
$JOB  PascalTest,username,,PascalTest.SysOut
Execution begun: Sat May 11, 1991 7:23 PM
;This job compiles, links, and runs a Pascal Program
$RUN  [Sys]<Sys>PascalFE.Run,PascalTest.Pas

Termination status code: 0

$RUN  [Sys]<Sys>Linker.Run,PascalTest.Obj,PascalTest.Run

Termination status code: 0

$RUN  PascalTest.Run

Termination status code: 0

$END

Execution ended: Wed Sep 23, 1992 7:25 PM
```

Also in the foreground partition:

- Batch Manager processes only one batch job, rather than processing a queue of jobs.
- A job can be cancelled by pressing **ACTION-FINISH**.
- Batch will set the path and prefix for a job using information from the .user file specified in the job statement.
- In a nonsecure environment (that is, non-Access Control), the user name specified in the job statement replaces the user name you are signed in as, even after Batch execution is complete.

  Thus, if you sign on to the system as User1, but specify User2 in a Job statement, when Batch execution is complete, you are signed on as User2 and must logout if you wish to return to your original user name.
- Keyboard bytestreams return data from the actual keyboard instead of redirecting the bytestream to read data embedded in the JCL file (there is no SysIn bytestream facility).

- Video bytestream output for each job step appears on the display. Batch Manager also echoes processed JCL statements to the SysOut file when one has been specified (refer to Example-3).
- Batch Manager ignores all **Log** statements and does not maintain a batch log file.
- The SysOut output can be sent to a disk file or a printer spooler file by specifying either one in the following **Job** statement:

  $JOB jobname,username,[password],[SysOut file]

Note: For a job executing in the foreground partition, Batch searches for [Sys]<Sys>Username.user. If a username is missing from a Job statement, Batch defaults to ***[Sys]<Sys>.user****; if this file is absent, Batch suffers an error.*

The following is a typical sequence for batch processing in foreground:

1. A JCL file is created or edited using the Editor, a text processor, or the Command File Editor's output to a JCL file
2. Batch Manager is invoked using the **Batch Foreground** command; the JCL file is specified for processing.
3. Batch Manager processes all job steps contained in the JCL file and terminates.

Recovering Foreground Batch Jobs

If a foreground Batch job execution aborts for any reason, before attempting to restart the job:

1. Check the SysOut file (if one was specified) to determine which job step was executing.
2. As appropriate, make required changes in the JCL file.

Use the **Batch Foreground** command to restart the job.

Normally, execution restarts at the beginning of the file. If you insert a **RestartLabel** statement within the job and specify **Yes** in the **Restart JCL File?** field of the **Batch Foreground** command, you can restart execution at the last processed **RestartLabel** statement. Refer to **RestartLabel** in Section 6 for more information.

Batch Processing in a Background Partition

In a background partition with Queue Manager installed, first install the SysIn and SysOut bytestreams using the **Install BS Batch** command. (Since Install BS Batch is a system service, it must be installed from the primary partition.) Then install Batch Manager using the **Install Batch** command (refer to Section 4). To add a job entry to a batch queue, invoke the **Batch** command and specify an installed queue in the Queue parameter on the command form (refer to Section 4).

Batch Manager repeatedly polls Queue Manager to obtain a queue entry for processing. Batch Manager marks this queue entry for use, and deletes it after processing the job unless a **Repeat After Time** is specified. This marked entry is not available to any other Batch Manager.

The **Batch Status** command (refer to Section 4) is used to monitor and cancel any background batch job.

Background processing does not permit direct interaction with Batch Manager through the keyboard or video display; instead, a SysIn bytestream redirects keyboard bytestreams to read data embedded in the JCL file. For commands that require confirmation, the JCL writer is responsible for the entry information. For example, the yes/no confirmation to overwrite a file with the **Copy** command must be provided in the JCL file.

Also consider that the SysIn bytestream:

- can only be directed from the current JCL file
- cannot have imbedded keyboard codes (for example, **GO** and **FINISH**)
- can be terminated in the JCL file with the **End** statement

A SysOut bytestream redirects video outputs to either a disk file or printer spooler.

While Batch Manager loads and activates each job step within the JCL file (refer to Example-1), the SysOut bytestream remains open until all job step output appends to the SysOut file (refer to Example-3). The SysOut file name is specified in the following **Job** statement:

$JOB jobname,username,[password],[SysOut file]

Note: For a job executing in the background partition, Batch searches for [Sys]<Sys>Username.user. If a username is missing from a Job statement, Batch defaults to [Sys]<Sys>.user; if this file is absent, Batch suffers an error.

The default SysOut file for a job executing in the background partition is:

[Scr]<Batch>JCLFileName.sysout$$Date-Time.log

The **[Scr]<Batch>** directory must exist for Batch to generate the default SysOut file.

Jobs that install system services, accept direct keyboard input, or use the Video Access Method (VAM) or Video Display Management (VDM) cannot be invoked in background batch (refer to either the *BTOS II System Reference Manual* or the *CTOS Operating System Concepts Manual* for more information).

Batch Log File

Batch Manager maintains a sequential log file of all jobs processed to date. If a batch log file exists, Batch Manager appends information to it; if a log file does not exist, Batch Manager creates a new log file at installation time. Information recorded includes the time Batch Manager was installed, each job's Job statement, start and finish times, and the termination status code for the last job step executed.

The **[Scr]<Batch>** directory contains all batch log files. The entry format is **xxyy.Log**, where **xx** is the batch scheduling queue served by Batch Manager and **yy** is the Batch Manager identification number. For example, a Batch Manager with identification number 00 serving a Batch scheduling queue named BATCH would maintain a log file named **[Scr]<Batch>BATCH00.Log** .

Because Batch Manager opens batch log files in long-lived mode, you cannot directly access the files through the operating system or an application; however, you can print a batch log file by using the **Batch Status** command (refer to Section 4). A job can also write to the log file with the Log JCL statement.

The batch log file is directed to the print queue **[Spl]** to be printed. However, if **[Spl]** is not the default printer, the log files are stored in the **[Scr]<Batch>** directory. Once you print, you start a new Batch log.

Recovering Background Batch Jobs

If, during the course of executing a background batch job, the system crashes or is reset, a valid job entry remains in the Batch Scheduling Queue. Rebooting the system and installing Queue Manager and Batch Manager causes this partially executed job to begin again.

If this is not desirable, perform one or more of the following actions before installing Queue Manager and Batch Manager:

1. If your Batch file contains **Log** statements, check the Batch Log file to determine which job was executing
2. Check the SysOut file to determine which job step was executing. If a SysOut file was not specified, check the **[Scr]<Batch>** directory.
3. As appropriate, make required changes in the JCL file.
4. Delete all batch queue files (for example, **[Scr]<Batch>*.queue**) to start over.

Section 6
Job Control Language Statements

This section describes the purpose and proper use of Job Control Language (JCL) statements to direct batch processing operations. It includes rules for statement syntax and for optional and required statement parameters.

JCL statements control run time operations such as:

- establishing job and user names
- loading and activating run files
- passing parameters to run files
- declaring run time variables

The Job Control Language statements are presented in alphabetical order in this section. Each presentation includes a functional description, the format of the statement, required and optional parameters, if any, and one or more examples.

Job Control Language Syntax

JCL syntax comprises the rules for using metacharacters, batch variables, and control statements. Because a JCL syntax error will always result in aborting the batch job, this section is critical for the successful use of Batch Manager JCL files.

Metacharacters

Metacharacters separate statement parameters or signal a new statement, function, continued line, comment, or literal character. Table 6-1 presents each metacharacter, along with a functional description and example.

Table 6-1. JCL Metacharacters

Metacharacter	Function/Example
,	Commas separate statement parameters; two adjacent commas denote a null parameter. Example: $Run FileA.Run p1,p2,,p4
( )	Parentheses enclose nested parameters and define order of operations. Example: $Run FileA.Run,(subparam1,subparam2),p2,p3 $If (x or y) and (t or w)
$	A dollar sign signals the start of a statement. It is optional. Example: $Cancel or Cancel
&	The ampersand indicates a statement or function that continues on the next line. Example: $Run [Sys]<sys>Example.Run,p1,p2,p3,p4,& p5,p6
;	A semicolon reserves the remainder of the line for a comment. The semicolon applies to everything that follows it on the line; to continue the comment, you must begin the next line with a new semicolon. Example: $Log 'End of FileA' ;message to log ;file

continued

Table 6-1. JCL Metacharacters (cont.)

Metacharacter	Function/Example
\	A reverse virgule precedes a metacharacter symbol that must be interpreted literally. Example: $Run AddMoney.Run,\$40.00, \$20.00
'	Pairs of single quotation marks enclose text that is to be interpreted as a literal. Example: x = '\' Foo*\" ;results in: ' Foo* ' var = '' ; null string
%	The percent sign precedes a number that identifies a parameter passed to a JCL file (%n). Example: Command **BATCH FOREGROUND** **JCL File** **Test.jcl** **[Parameters]** **'[sys]<sys>.user'** **[Restart jcl file?]** Line in a JCL File: $Command Copy, '%0', copytest In the example, 0 is a placeholder referencing the passed parameter '[sys]<sys>.user'. When using a placeholder for a string parameter, be sure to enclose the string in quotes both on the parameter line and in the JCL file.

continued

Table 6-1. JCL Metacharacters (cont.)

Metacharacter	Function/Example
% (continued)	The percent sign can also indicate an external variable (%variable), which is always a string. Examples: /* External variable being read */ $Assign GlobalVar, %ExtVar GlobalVar = StrToNum(%ExtVar) GlobalVar = GlobalVar + LocalVar /* External variable being assigned */ %ExtVar = NumToStr(GlobalNum) %ExtVar = CopyString(LocalString) %ExtVar = SomeVar $Assign $ExtVar, SomVar /* External variable being written */ %ExtVar = ConcatStrings(%ExtVar, Variable) %ExtVar = %ExtVar + GlobalVar /* External variable being deleted */ %ExtVar = '' $Deassign(%ExtVar)

System Constants

Batch Manager maintains a collection of system constants that provide information to JCL files for use in conditional statements. They cannot be modified by JCL statements.

Table 6-2 contains a list of the system constants and their values.

Table 6-2. System Constants

Constant	Value
HardwareID	existing hardware identification on the system executing Batch; default is zero (Batch calls ReadHardID once when it starts the JCL file and does not call it again, even if the JCL file executes the SetHardwareID procedure.)
KeySwitch	0 = not SRP 1 = manual 2 = remote 3 = normal
MemAvail	numeric (Kb) tota;l amount of memory on the system (for example, a 4096 Kbyte system)
nFP, nSP, nTP, nDP, nCP, nGP	the number of each board on SRP (If cp00 and cp01 are present, then nCP = 2.)
OSRelease	numeric; version of the operating system (CurrentOSVersion() is described in the *CTOS System Procedural Interface Reference Manual.*)

continued

Table 6-2. System Constants (cont.)

Metacharacter	Function/Example
OSRevision	numeric; revision of the operating system (CurrentOSVersion() is described in your *CTOS System Procedural Interface Reference Manual.*)
Processor	0 = 186 1 = 286 2 = 386
RunningOnSrp	0 = false 255 = true
SRPServer	0 = false 255 = true
SystemType	0 = Stand-alone 1 = Cluster 2 = Cluster Local File System 3 = Server 4 = LFS Booted From Server

Table 6-3 contains a list of the system constants and values used by the Installation Manager during software installation.

Table 6-3. Installation Manager System Constants

Constant	Value
InstallType	0 = floppy 1 = tape 2 = Server 4 = Software Distribution
Public	0 = false 255 = true
Unattended	0 = false 255 = true (If true, default variables are used and you are not prompted during installation.)

Batch Variables

Batch Manager allows numeric and string variables. Batch numeric variables, internally represented as DWord, are user-defined, unsigned numeric values (decimal or hexadecimal). A batch variable name must begin with an alphabetic character (a ... z, A ... Z), and may not exceed 12 characters.

Examples:

- $Assign x, 10 * 57 -or- x = 10 * 57

  Result is: 570

- $Assign y, 123abcefH -or- y = 123abcefH

  Result is: 305839343

- $Assign z, 'This is a test. -or- z = 'This is a test.'

  Result is: This is a test

- $Assign BritishPound, '\\0B1h'

  Result is: £

- $AssignLocal locvar, 255

  Result is: 255

You can use negative numeric values in mathematical equations up to a maximum value of -65536; however, Batch cannot display negative numbers or test conditional expressions that contain negative numbers.

Batch variables can be global or local in scope. Global variables can be amended by all the embedded JCL files. Local variables can be amended only in the JCL file in which they are declared, using the AssignLocal statement. Once declared, a global batch variable remains available until the job terminates or the variable is deassigned using Deassign; whereas, a local batch variable remains available until the JCL file in which it was declared terminates, or it is deassigned using DeassignLocal.

You can define up to 64 global variables in a Batch suite. A maximum of 6 KB is available for string assignments. For each JCL file, you can define up to 64 local variables.

If you define a local and global variable with the same name, Batch uses the local variable. For example:

```
$AssignLocal A, 'This'
$Assign A, 10
$Assign B, 10
```

Later, if Batch had to solve A + B, it would add 'This' + 10, an unworkable math operation.

You can use batch variables as parameters in **Run** and **Call** statements, as numbers, and as numeric constants wherever numeric variables are legal. For example:

```
x = 15
$command SetProtection,myfile,x
```

Batch Manager supports integer arithmetic using previously initialized numeric variables and numeric constants. Batch Manager also supports conditional expressions. Expressions may include nested parentheses.

Table 6-4 lists the arithmetic operators Batch Manager supports.

Table 6-4. Arithmetic Operators

Order of precedence	Operator	Symbol	Description
1	NOT		Unary negation
2	*		Multiplication
3	/		Division
4	MOD		Modulo
5	+		Addition
6	-		Subtraction

continued

Table 6-4. Arithmetic Operators (cont.)

Order of precedence	Operator	Symbol	Description
7	EQ	=	Equal to
8	NE	<>	Not equal to
9	GT	>	Greater than
10	LT	<	Less than
11	GE	>=	Greater than or equal to
12	LE	<=	Less than or equal to
13	AND		Logical AND
14	OR		Logical OR
15	XOR		Logical XOR

Batch Manager supports 2K strings in all string manipulation functions. String expressions can be tested for equality and inequality by using EQ (=) and NE (<>).

Expressions with numeric and string subexpressions are allowed. For example, $IF (nVar <= 10) and (sTemp = 'test').

Note: *Constructs and operators are reserved words; they should not be used as variable names.*

External Variables

Batch Manager also supports external variables, which begin with a percent sign (%). The name of an external variable must begin with a letter of the alphabet and, unlike local and global variable names, can exceed 12 characters.

Example:

```
%NuVol.NuDir.NuFile = '[Sys]<Nu>NuConfig.Sys'
$Display(%NuVol.NuDir.NuFile)
```

Result:

```
[Sys]<Nu>NuConfig.Sys
```

External variables can also include standard passed parameters to pass a name of an external variable to Batch.

Example:

Executive command line:
Batch Foreground
JCL File **Job.jcl**
[Parameter] **UserName**
[Restart LCL File?]

JCL contents:

```
MyName = 'Bob'
if (MyName NE %%0)
        cancel
EndIf
```

This will look for an external variable called %UserName, that is,

```
If (MyName NE %UserName)
```

The name of an external variable cannot include an arithmetic operator or a colon, which is reserved for use after Batch labels. Both of the following JCL statements will result in syntax errors:

```
%*BadVar = '[Sys]<Sys>'
%BadVar:
```

If, instead of preceding the name with a percent sign (%), you embed the percent sign (%) within the variable name, the name will not represent an external variable. It will become a global or local variable. Here are two examples:

```
Global%x = 10;          Global variable
$AssignLocal Local%y;   Local variable
```

Restrictions of External Variables:

The following restrictions exist for external variables:

- You cannot use external variables (or other kinds of variables) to name run files or commands in **$Run** or **$Command** statements. However, parameters can be external variables.
- JCL statements that require local variables (**$DisplayLocal**, **$AssignLocal**, **$GoTo**) cannot use external variables.
- Batch labels cannot be external variables.

GoTo Labels

A **GoTo** label is a token followed by a colon.

Batch **GoTo** labels are used for transferring control with **GoTo** statements (see the example in **GoTo**). A **GoTo** label is treated as a local variable. It must begin with an alphabetic character and cannot exceed 12 characters. For example:

```
step1:
.
.
.
step2:
```

Shared Resource Processor Board Name Labels

SRP board name labels (for example, GP00 or CP02) direct JCL constructs to be executed on specified shared resource processor boards. These labels are used in combination with **$EndBoard** statements to bracket JCL constructs. For example:

```
$FP00
  $Command...
  $Run...
  $RunNoWait...
  $Call...
  $CallNoWait...
  $End...
$EndBoard...
```

Control Variables

Batch Manager also provides predefined and preset variables. These control (or system) variables can be modified in a JCL file.

Table 6-5 contains a list of these variables.

Table 6-5. Control Variables

Variable	Description
Date	current date expressed as a string using DtTemplate to determine the format
DtTemplate	template number (0 through 17) for the Date variable; default is 5 (for example, Tue Sep 15, 1992 10:05AM); each value gives a different date/time form (Refer to the *CTOS System Reference Manual* for details on Native Language Support Templates.)

continued

Table 6-5. Control Variables (cont.)

Variable	Description
DumpValue	debugging aid for JCL writers; when set to True (any non-zero value), Batch displays all variable content in parentheses next to the variable; when set to False (zero, the default), Batch displays an asterisk before the first executed statement within an IF or WHILE statement.
Erc	status code returned from last Run or Command
ErcMsg	status message returned from last Run or Command, if returned; placed in ACB.pbercMsgRet
VideoLevel	value set by the statements EchoOn (0 is default), EchoSome (1), and EchoOff (2); determines how much information will be displayed as a JCL file is processed

Table 6-6 contains a list of variables the Installation Manager uses during software installation.

Table 6-6. Installation Manager Control Variables

Variable	Description
CmConfigFile	name of current Context Manager configuration file. Default = blank.
CmdFileTo	command file where commands are to be added. The default is the current command file [Sys]<Sys>Sys.cmds.
CmdFileFrom	default = [Scr]<$>Install.cmds
DeviceFrom	device from which the installation is being performed; dependent on value of InstallType (for example, if InstallType = 0, then DeviceFrom = '[f0]', '[f1]', and so on.). Default = blasnk If Installtype = 4, (Software Distribution) Device From contains the path specification and a file prefix indicating the Style ID and version number (for example, [Sys]<SWD-Hold>B25-EM>2.2.3>).
DirectoryTo	destination directory for software installation; passed to Batch from Installation Manager. Default = blank.
MsgFile	the name of the message file associated with the subpackage. This is used as an argument to the keyword InitMsgFile. Default = blank.
Pkgs	names of selected subpackages; SubString function queries this variable; passed to Batch from Installation Manager. Default = blank
VolumeTo	name of selected volume onto which software installs; passed to Batch from Installation Manager. Default = blank.

Job Control Language Syntax Errors

Any language syntax error in a JCL file will always cause the batch job to terminate whether or not **ContinueOnError** is set.

Assign

This statement instructs Batch Manager to declare a global batch variable by assigning, to a variable name, a numeric value (decimal or hexadecimal), an arithmetic expression reducing to such a value, or a string (refer to Batch Variables).

Every variable must be declared before it can be part of an expression. String variables must be enclosed in single quotes and hex number assignments must end with h or H.

Format: $Assign VariableName, {value, expression, string}

Examples:
```
$Assign var1, 20
$Assign var2, var1/2
$Assign var3, 'anystring'
$Assign BritishPound, '\\0B1h'
$Assign hexnum, 123abcefH
$Assign negx, -15
$Assign x, var1 + negx
```

Results:
```
var1 = 20
var2 = 10
var3 = anystring
BritishPound = £
hexnum = 305839343
x = 5
```

A null string can be assigned by placing two single quotes together: ''. The string length of a null string is 0.

You cannot display negative numbers.

The maximum number of global variable items allowed is 64. The maximum numeric variable value is 4,294,967,295.

You can also assign variables without the **Assign** statement by using the following:

Format: VariableName = {value, expression, string}

Examples:
```
var1 = 20
var2 = var1/2
var3 = 'anystring'
BritishPound = '\\0B1h'
hexnum = 123abcefH
negx = -15
x = var1 + negx
```

Results:
```
var1 = 20
var2 = 10
var3 = anystring
BritishPound = £
hexnum = 305839343
x = 5
```

AssignLocal

This statement instructs Batch Manager to declare a local batch variable in the same manner as the **Assign** statement, but this variable is local to the current JCL file only and will be deassigned upon termination of the JCL file where it is declared or upon explicit deassignment, using **$DeassignLocal**.

Format: $AssignLocal VariableName, {value, expression, string}

Examples:
```
$EchoOff
GlobalNum = 2
$AssignLocal var1, 20
$AssignLocal var2, var1/2
$AssignLocal var3, 'anystring'
$AssignLocal var4, GlobalNum + 4
$AssignLocal BritishPound, '\\0B1h'
$DisplayLocal (var1)
$DisplayLocal (var2)
$DisplayLocal (var3)
$DisplayLocal (var4)
$DisplayLocal (BritishPound)
$End
```

Results:
```
var1 = 20
var2 = 10
var3 = anystring
var4 = 6
BritishPound = £
```

BroadcastMessage

This statement enables you to send a mail message to all users in your mail center. OFIS Mail must not be running in another partition when this statement runs. You must have your OFIS Mail user ID keyword in your user file (for example, :MailUserName:J.Doe).

For an attachment, you must specify a fully qualified file name.

Format: $BroadcastMessage (message[, subject][, attachment])

where: **message** is the text of the mail message

subject is optional. The default is: "Message sent from a JCL file"

attachment is optional

Example:
```
$Assign x, '1.0 version of software installed'
$Assign y, 'software installation'
$Assign z, '[Sys]<Sys>Test.file'

$BroadcastMessage (x, y, z)
```

Call

This statement instructs Batch Manager to:

- suspend processing of the calling JCL file and start processing the called JCL file
- ignore the Job statement in the called JCL file
- allow JCL file nesting
- resume the calling JCL file, after the called JCL file is processed, starting with the JCL statement after the Call statement

Format: $Call filespec[, parameters]

where: **filespec** is either a partial or a full file specification. If a partial specification is used, the volume and directory defaults will be those specified by the job card user name file or a **$Path** statement.

parameters are the program parameters to be passed to a called JCL file: p0, p1, ..., pn (where p0 refer to %0, p1 refers to %1, and pn refers to %n).

Example: $Call Bob.JCL, p0, p1, p2, p3, p4, p5

where: **Bob.JCL** is the JCL file called and processed;

p0...p5 referred to as **%0...%5** in a called JCL file.

*Note: Subparameters cannot be passed with a **Call** statement.*

If you attempt to call JCL files that reside on a write-protected diskette, an error will result. First copy the files to **[Scr]<$>filename** or **[Sys]**; then call them.

CallNoWait

This statement, specifically for the XE shared resource processor, instructs Batch Manager to continue processing after starting another JCL file on another board. Batch processing on the current board does not wait for a response from the called JCL file.

Format: $CallNoWait filespec[, parameters]

where: **filespec** is either a partial or a full file specification. If a partial specification is used, the volume and directory defaults will be those specified by the job card user name file or a **$Path** statement.

parameters are the program parameters to be passed to a called JCL file: p0, p1, ..., pn (where p0 refers to %0, p1 refers to %1, and pn refers to %n).

Example: $CallNoWait Bob.JCL, p0, p1, p2, p3, p4, p5

where: **Bob.JCL** is the JCL file called and processed;

p0...p5 refer to parameters (%0...%5) that are to be passed to file **Bob.JCL.**

Note *Subparameters cannot be passed with a* ***CallNoWait*** *statement.*

Cancel

This statement instructs Batch Manager to cancel the current job. A called JCL file can issue a **Cancel** statement, but the batch job will not return to the calling JCL file.

Format: $Cancel

CancelOnError

This statement instructs Batch Manager to cancel processing the current job when a job step terminates with a nonzero error code. This is the default setting in a JCL file. This default can be changed by using a **ContinueOnError** statement.

You can use this statement with Erc, ErcMsg, or DisplayErrorMessage to control the JCL processing based upon error messages and error codes returned.

Format: $CancelOnError

Example:
```
$CancelOnError
$Run Filename.run
```

Result: If Filename.run returns a non-zero value, then Batch cancels this job.

Command

This statement instructs Batch Manager to invoke one of the commands defined in the user's current command file. Using the **Command** statement serves as an alternative to invoking the **Run** statement to start an Executive command.

You must execute all Executive intrinsics using **Command**, because there are no run files associated with them. To determine if a command is an Executive intrinsic, type the command name on the Executive command line, press **RETURN**, and then press **HELP**. If the command is an Executive intrinsic, the **Run File** field will display an exclamation point (!) followed by a number. For example, if you type **Copy** on the Executive command line and press **RETURN**, !2 displays in the Run File field, identifying the **Copy** command as an Executive intrinsic command.

When executing in a background partition, you must supply the yes/no parameters in your JCL file for commands that require user confirmation (for example, **Copy** or **Rename**).

Format 1: $Command command name [, parameters]

Format 2: $Command command name, (subparam1, subparam2), param2

where: **parameters** are the program parameters to be passed: p1, p2, ..., pn.

Examples:
```
$Command VID, y

$Command FILES, '*'

$Command COPY, '*', '*>save'

$Command FILES, (foo1, foo2,foo3), yes

$Command LINKER, @obj.fls, Test.Run

$Command Set Protection, MyFile,15
```

Program parameters may include the wild card characters question mark and asterisk (? and *) and the at-sign (@) character. Wild cards can be expanded by either Batch or the Executive. Batch wild card expansion is functionally equivalent to Executive wild card expansion.

If the command expression is not expanded by the Executive, and you want the command (that you are trying to execute) to expand the wild cards, then the statement must be enclosed in quotes; however, if you want Batch to do the expanding, leave the quotes off.

In the examples below, the first option of enclosing the **Files** command in quotes is the best choice. This practice allows Files.run to handle the memory management associated with the expansion of wild cards.

- If you use quotes (Command Files, '[Sys]<*>*'), batch simply passes what is in the quotes to the **Files** command
- If quotes are left off (Command Files, [Sys]<*>*), batch expands the wild card and passes the expanded list to the command being executed.

You can determine whether the Executive will automatically expand a wild card in one of two ways:

- type the command at the Executive using wildcards in the parameters and press **RETURN** to see if the wildcard automatically expands; if it does, then you can let Batch do the expanding by leaving the quotes off.
- Access the command form using the **Edit** function of the Command File Editor and check for asterisks. If the command form contains asterisks, the Executive automatically expands wild cards.

Caution:

Avoid using the following statement in a JCL file:

Command Delete. [sys]<$*>*

This command deletes the Batch context file and causes Batch to abort once the **Delete** command executes.

ConcatStrings()

This statement concatenates two strings and assigns the result to a third string. The maximum concatenated string size is 2048 characters.

Format: str = ConcatStrings(str1, str2)

Example:
```
str1 = 'key'
str2 = 'board'

str = ConcatStrings(str1, str2)

$Display (str)
```

Result: str = keyboard

ContinueOnError

This statement instructs Batch Manager to continue processing the current job even if a job step terminates with a nonzero error code. If **ContinueOnError** is not used, **CancelOnError** is the default.

A syntax error in a JCL construct always cancels the current job, even if **ContinueOnError** is used.

Format: $ContinueOnError

Example 1:
```
$ContinueOnError
$Run [sys]<sys>MaillServer.run
$Run [sys]<UTE>UniscopeGateway.run,1
```

Result: Since the Mail Server run file name is misspelled, the Mail Server is not installed, but Batch continues the job and installs the Uniscope Terminal Emulator.

Example 2:
```
$ContinueOnError
$Dspline('Hello')
$Run [sys]<sys>MailServer.run
```

Result: Since DisplayLine is misspelled, this job terminates with an ERC 5, and the Mail Server is not installed.

CopyString()

This statement copies the contents of one string to another.

Format: x = CopyString (string, index, length)

where: **string** is an alphanumeric sequence

index is the position to start the copy (beginning at 0)

length is the number of characters to copy

Example: y = 'This is a string'

x = CopyString (y, 10, 6)

Result: Copies the 11th index (where index begins with zero) in y into x for 6 characters

x = string

Deassign

This statement instructs Batch Manager to deassign the specified global variable.

Format: $Deassign (VariableName)

Example: Hex.var = 0A12h

$Deassign (Hex.Var)

DeassignLocal

This statement instructs Batch Manager to deassign the specified local variable.

Format: $DeassignLocal (LocalVarName)

Example: $AssignLocal Hex.Var, 0A12h

$DeassignLocal (Hex.Var)

DeviceType

This statement takes a device name as an argument and returns the numeric device type, allowing you to determine the device type from which you will install software.

Format: type = DeviceType(devicename)

where: **type** is one of the following numeric codes:

0	Unknown device
1	5.25-inch floppy (low density)
2	5.25-inch floppy (high density)
3	3.5-inch floppy (low density)
4	3.5-inch floppy (high density)
5	Hard disk

Examples:
```
type = DeviceType('f0')

devicename = 'd0'

type = DeviceType(devicename)
```

Display

In foreground operation, this statement instructs Batch Manager to display the specified output to video and SysOut (when specified). In background operation, the output is written to the SysOut file.

Format: $Display (output)

where: output can be a global variable, a value, a numeric constant, or a string literal. (The literal must be enclosed in single quotation marks.)

Examples:
```
$EchoOff
$Display ('This is a display. ')
$Var1 = 15
$Display (var1)
$Display (3500)
$Display (' var1 = ', var1)
$Var2 = var1 + 30
$Display (' var2 = ', var2)
```

Results: This is a display. 153500 var1 = 15 var2 = 45

$Display accepts only assigned variables, not expressions, as arguments. To display an expression, first assign the expression to a variable; then use the variable as the argument. For example:

```
y = (a + b)

z = (c + d)

$Display (y, z)
```

Note: *To display local variable values, use* ***$DisplayLocal.***

DisplayAndWait

In foreground operation, this statement instructs Batch Manager to display text and then wait for any keystroke from the user before continuing.

Format: $DisplayAndWait (value1[, value2[, value3[, ...]]])

where: **valueN** can be a global variable, a value, a numeric constant, or a string literal. (The literal must be enclosed in single quotation marks.)

Examples:
```
$EchoOff
$Assign Msg1,'This application requires request
codes.\\0Ah'
$Assign Msg2,'Please reboot your system after installation is
complete.'
$DisplayAndWait (Msg1, Msg2)
```

Results:
```
This application requires request codes.
Please reboot your system after installation is complete.
```

$DisplayAndWait accepts only assigned variables, not expressions, as arguments. To display an expression, first assign the expression to a variable; then use the variable as the argument. For example:

```
y = (a + b)

z = (c + d)

$DisplayAndWait (y, z)
```

Note: *To display local variable values, use* ***$DisplayLocal.***

DisplayErrorMessage

This statement (supported in foreground only) causes a Native Language Support (NLS) message and/or a JCL user-defined message to be displayed in a window

The window can display a maximum width of 76 characters.

Format: $DisplayErrorMessage (erc)

or

$DisplayErrorMessage (erc, [message, fBoth])

where: **fBoth** TRUE causes NLS and user-defined messages to display. True must be defined as 255.

fBoth FALSE causes a user-defined message to display only if an NLS message is not available. False must be defined as 0.

Example 1: NLS message only:

$DisplayErrorMessage (203)

Result: Message displayed in pop-up window:

No Such File (Error 203)

Example 2: NLS message User-defined message:

$Assign FileName, 'file1'
$DisplayErrorMessage (203, FileName, TRUE)

Result: Message displayed in pop-up window:

No Such File (Error 203)
file1

Example 3: NLS message unavailable:

```
$Assign MSG, 'Message is not in NLS msg file'
$DisplayErrorMessage(32946, MSG, 0)
```

Result: Message displayed in pop-up window:

Message is not in NLS msg file

Example 4: NLS message available:

```
$DisplayErrorMessage(203, MSG, FALSE)
```

Result: Message displayed in pop-up window:

No such file (Error 203)

Since FBoth is FALSE, the user defined message, MSG, is not displayed.

DisplayLine

In foreground operation, this statement instructs Batch Manager to display the specified output to video and SysOut (when specified). This statement is the same as the **Display** statement except it forces a carriage return, and line feed after displaying the requested text.

In background operation, the output is written to the SysOut file.

Format: $DisplayLine (value1[, value2[, value3, ...]]])

where: **valueN** can be a global variable, a value, a numeric constant, or a string literal. (The literal must be enclosed in single quotation marks.)

Examples:
```
$EchoOff
$DisplayLine ('This is a display. ')
$var1 = 15
$DisplayLine (var1)
$DisplayLine (3500)
$DisplayLine ('var1 = ', var1)
$var2 = var1 * 2
$DisplayLine ('var2 = ', var2)
$End
```

Results:
```
This is a display.
15
3500
var1 = 15
var2 = 30
```

Note: *To display local variable values, use the* ***DisplayLocal*** *statement.*

DisplayLocal

In foreground operation, **DisplayLocal** instructs Batch Manager to display the specified output to video and SysOut file (when specified). In background operation, the output is written to the SysOut file.

Format: $DisplayLocal (output)

where: **output** can be a local variable, a value, a numeric constant, or a string literal. (The literal must be enclosed in single quotation marks.)

Examples:
```
$AssignLocal local.var, 'This is a local variable'
$DisplayLocal (local.var)
```

Results: This is a local variable

Dump

Dump instructs Batch to list all batch variable names and their associated contents that are encountered from the beginning of the file to the **Dump** statement. System constants and control variables are also dumped.

All local variables, when present, are preceded by an asterisk (*) and are dumped first.

Format: $Dump

Result:
```
* scrn = 0
* temp = 0
* term = 0
RunningOnSrp = 0
SrpServer = 255
erc = 0
ercMsg =
SystemType = 3
Processor = 2
HardwareId = 0
MemAvail = 4096
DeviceFrom =
OSRelease = 11
OSRevision = 7
InstallType = 0
CmConfigFile =
Keyswitch = 0
```

```
Date = Wed Sep 16, 1992 5:42 PM
DtTemplate = 5
Public = 0
CmdFileFrom = [scr]<$>install.cmds
MsgFile =
DirectoryTo =
CmdFileTo = [sys]<sys>sys.cmds
VolumeTo =
VideoLevel = 0
DumpValue = 0
Unattended = 0
nFP = 0
nTP = 0
nCP = 0
nSP = 0
nDP = 0
nGP = 0
```

EchoOff

This is a video level statement (refer to Table 6-3) that suppresses all video output except that which is displayed using the following statements in the JCL file:

Command
Display
DisplayAndWait
DisplayErrorMessage
DisplayLine
DisplayLocal
Run

It sets the Batch predefined control variable VideoLevel to 2.

Format: $EchoOff

Example:
```
$JOB EchoOff,,,[Sys]<Sys>EchoOff.out
$EchoOff
$Command Version, [Sys]<Sys>Batch.run
$DisplayLine(erc)
$End
```

Result:
```
JOB EchoOff,,,[Sys]<Sys>EchoOff.out
Execution begun:  Mon Sep 7, 1992 4:11 PM
$EchoOff

Run File Version Program 12.2.0

[Sys]<Sys>Batch.run      12.2.0
0
```

EchoOn

This is a video level statement (refer to Table 6-3) that displays all Batch Manager output. This is particularly helpful when debugging JCL file processing. **EchoOn** is the default.

It sets the Batch predefined control variable VideoLevel to 0.

Format: $EchoOn

Example:
```
$JOB EchoOn.jcl,,,[Sys]<Sys>EchoOn.out
$EchoOn
$Command Version, [Sys]<Sys>batch.run
$DisplayLine(erc)
$end
```

Result:
```
JOB EchoOn.jcl,,,[Sys]<Sys>EchoOn.out
Execution begun:  Mon Sep 7, 1992 4:14 PM
$EchoOn
$Command Version, [Sys]<Sys>batch.run

Run File Version Program 12.2.0

[Sys]<Sys>Batch.run:  12.2.0

Termination status code:  0

$DisplayLine(erc)
0

$end

Execution ended:  Thu Jun 4, 1992 4:14 PM
```

EchoSome

This is a video level statement (refer to Table 6-3) that suppresses all Batch Manager output except for video output of applications running under Batch Manager and the following statements:

Command
Display
DisplayAndWait
DisplayErrorMessage
DisplayLine
DisplayLocal
Run

It sets the Batch predefined control variable VideoLevel to 1. EchoSome and EchoOff are nearly the same. The one difference is that EchoSome displays $Command and $Run lines, while EchoOff does not.

Format: $EchoSome

Example:
```
$JOB EchoSome.jcl,,,[Sys]<Sys>EchoSome.out
$EchoSome
$Command Version, [Sys]<Sys>batch.run
$DisplayLine(erc)
$end
```

Result:
```
JOB EchoSome.JCL,,,[Sys]<Sys>EchoSome.out
Execution begun:  Wed Jun 3, 1992 4:03 PM
$EchoSome

$Command Version , [Sys]<Sys>batch.run

Run File Version Program 12.2.0

[Sys]<Sys>batch.run:    12.2.0

0
```

Else

This is an optional statement in the $If...$Else...$EndIf construct. **Else** affects the extent of skipping and processing between **If** and **EndIf** as follows:

- From the JCL statement at which the **If** condition is satisfied, the program skips to **EndIf.**
- If the **If** condition is not satisfied before the program reaches **Else**, the program processes all statements from **Else** to **EndIf.**

Refer to **If** and **EndIf** for more information.

The maximum number of statement nesting is 12.

Format:

```
$If (conditional expression)
.
.
.
$Else
.
.
.
$EndIf
```

Example:

```
$If X EQ 10
   $DisplayLine ('X is equal to 10')
   $DisplayLine ('This is the limit')
$Else
   $DisplayLine ('X is not equal to 10')
   $DisplayLine ('X is equal to', X)
$EndIf
```

End

This statement instructs Batch Manager to note the end of the current JCL file. End should be the last statement in a JCL file.

Format: $End

EndBoard

This statement, specifically for the XE shared resource processor, brackets processor board names and makes it clearer to the user when installing applications on different boards (for example, between TP00 and CP00).

For an example JCL file, refer to appendix C.

Format: $Endboard

Example:
```
TP00
   $run test1
   $run test2
$Endboard

$run extraprog

CP00
   $run test3
   $run test4
$Endboard
```

Result: Test1 and test2 run on the TP00 board, extraprog runs on the board this JCL file is executing on, test3 and test4 run on the CP00 board.

EndIf

This concludes a statement sequence begun by an **If** statement at the same nesting level. Each **If** statement must have a matching **EndIf** statement. (Refer to **If** and **Else** for more information.)

The maximum number of statement nesting is 12.

Format:

```
$If (conditional expression)
.
.
.
$Else
.
.
.
$EndIf
```

Example:

```
$If X EQ 10
   $DisplayLine ('X is equal to 10')
   $DisplayLine ('This is the limit')
$Else
   $DisplayLine ('X is not equal to 10')
   $DisplayLine ('X is equal to', X)
$EndIf
```

EndWhile

This concludes a statement sequence begun by a **While** statement at the same nesting level, according to the following rules:

- As long as the **While** loop condition is not satisfied, processing continues through each JCL statement and reaches the **EndWhile** statement in its turn.
- As soon as the **While** loop condition is satisfied, processing skips to the **EndWhile** statement.

The maximum number of statement that can be nested is 12.

There must be an **EndWhile** statement for each **While** statement. (Refer to **While** for more information.)

Format:
```
$While (conditional expression)
.
.
.
$EndWhile

.
.
.
```

Example:
```
$EchoOff
x = 1
$While x LE 5
   $DisplayLine ('x = ', x)
   x = x + 1
$EndWhile
```

Result:
```
x = 1
x = 2
x = 3
x = 4
x = 5
```

FileOpenStatus

This statement allows you to determine whether a file can be opened in a specific mode. **FileOpenStatus** takes the name of the file and the mode in which you intend to open it and returns an error code.

Format: erc = FileOpenStatus(filespec,mode)

where: **erc** defines the status of the JCL primitive

If the file can be opened in the specified mode, **erc** is zero. If not, **erc** is the appropriate file system error returned on the OpenFile call.

filespec is the filename

mode defines the mode in which the file is to be opened. The modes are the following:

mr moderead

mm modemodify

Examples: erc = FileOpenStatus('[Sys]<Nu>NuConfig.Sys>', 'mm')

moderead = 'mr'

modemodify = 'mm'

configFile = {Node A}'[Sys]<Nu>NuConfig.Sys'

runfile = '[Sys]<Nu>Nu.run'

erc = FileOpenStatus(configfile,moderead)

erc = FileOpenStatus(runfile,modemodify)

FileVersion()

This statement searches for a file and any version number associated with it.

Format: VerString = Fileversion(FileName)

where: **VerString** is a string containing a valid version, '255' or '0'.

'255' indicates that the file being checked exists, but the version number cannot be determined. For example, you may access a file that does not have a version number, the file may be in use, or you may not have access privileges for the file.

'0' indicates that the file does not exist or cannot be located.

FileName is a string literal or a string variable with the value of a file name.

Example 1:
```
FileName = '[Sys]<Sys>Batch.run'
VerString = Fileversion(FileName)
$DisplayLine(VerString)
```

Result: 2.3

Example 2:
```
VerString = Fileversion('[Sys]<Sys>Karen.User')
$DisplayLine(VerString)
```

Result: 255

This result indicates that [Sys]<Sys>Karen.User exists but it does not have a version attached to it.

Example 3:
```
VerString = Fileversion('[Sys]<Sys>NotHere.fls')
$DisplayLine(VerString)
```

Result: 0

This result indicates that [Sys]<Sys>NotHere.fls could not be found in the specified volume and directory path.

FrontPanel

This statement, specifically for the XE shared resource processor, changes the value on the front panel of the shared resource processor base enclosure. You can use it to determine how far a SysInit.jcl file has progressed. For example, you might change the panel to a new number after each service installs.

See Appendix C for a more detailed example of a sysinit file which uses the **FrontPanel** statement.

Format 1: $FrontPanel hh

Format 2: $FrontPanel Nvar or (Nvar)

where: **hh** is a hexadecimal (00 through FF) literal number.

Nvar or (Nvar) is a two-digit hex variable

Example:
```
$FrontPanel 1A
Nvar = 30h
$FrontPanel Nvar
Nvar = Nvar + 1
$FrontPanel (Nvar)
```

Result: First, 1A is displayed on the front panel.
Then 30 and, at last, 31 are displayed on the front panel.

GetMsg()

This statement retrieves a message from a message file and places the message in memory. Macros (for example %s) cannot be embedded in the message. You use **GetMsg** to conserve memory in a JCL file.

To conserve memory, use as few different variable names as possible for retrieving messages, or deassign unneeded variables.

Format: GetMsg(number)

where: **number** is a numeric literal or numeric variable

Example:
```
$EchoOff
$Assign file, '[Sys]<Sys>ercmsg.bin'
erc = InitMsgFile(file)
prompt = GetMsg(2)
$DisplayLine (prompt)
prompt = GetMsg(15)
$DisplayLine (prompt)
prompt = GetMsg(17)
$DisplayLine (prompt)
```

Result: End of medium (EOM) (Error 2)

No link block available (Error 15)

Mismatched response (Error 17)

GetMsg is used after InitMsgfile() has been executed. After executing InitMsgfile() once, you can execute GetMsg() any number of times. InitMsgfile() initialized a user-defined message file, ercmsg.bin, containing numbered messages. GetMsg() retrieves messages 2, 15, and 17. The variable prompt can now be used as a string to reference the message in any Batch construct.

Note: *If the message in the .bin file is longer than 2 Kbyte characters,* ***GetMsg*** *returns a null string.*

GoTo

This statement causes JCL file processing to jump (forward or backward) to a label and continue processing from the statement associated with the label.

Format: $GoTo label_name

Example:
```
$EchoOff
$GoTo LabelA

LabelB:
$Displayline('This is second')
$End

LabelA:
$Displayline('This is first')
$GoTo LabelB
```

Result:
```
This is first
This is second
```

If

This statement instructs Batch Manager to test a conditional expression, then process or skip JCL statements until it encounters an **Else** or **EndIf** statement. (Refer to **Else** and **EndIf** for more information.)

The maxiumum number of statement nesting is 12.

Format:

```
$If (conditional expression)
.
.
.
$Else
.
.
.
$EndIf
```

Example:

```
$If X EQ 10
   $DisplayLine ('X is equal to 10')
   $DisplayLine ('This is the limit')
$Else
   $DisplayLine ('X is not equal to 10')
   $DisplayLine ('X is equal to', X)
$EndIf
```

Implicit Call

This construct instructs Batch Manager to do the following:

- read a JCL statement as an implicit call construct
- call a JCL file without reading a Call statement
- process the file {local} [Sys]<Sys>file.jcl

where:

{local} [Sys]<Sys> is the default prefix, unless replaced by a **Prefix** statement.

file is the character string following the optional $ and terminated by an **EndOfLine.**

.jcl is the default suffix, unless replaced by a **Suffix** statement.

Refer to **Prefix** and **Suffix** for more information.

Format: $JCLfilename

Example 1: $Copy

is the equivalent to the following Call statement:

$Call [Sys]<Sys>Copy.jcl

Example 2:
```
$Prefix {USA}[NY]<Sales>
$Suffix .batchjcl
Report
```

is the equivalent to the following Call statement:

$Call {USA}[NY]<Sales>Report.batchjcl

InitMsgFile()

This statement opens a binary message file for retrieval of numbered messages. The message file must have been created in the Executive using the **Create Message File** command.

Format: InitMsgFile(MessageFilename)

where: MessageFileName is a string literal or string variable that references the file name and path of the message file.

Example:
```
$EchoOff
$Assign file, '[Sys]<Sys>ercmsg.bin'
erc = InitMsgFile(file)
prompt = GetMsg(2)
$DisplayLine (prompt)
prompt = GetMsg(15)
$DisplayLine (prompt)
```

Result: End of medium (EOM) (Error 2)

No link block available (Error 15)

Job

The Job statement must be the first statement in a JCL file (unless this JCL file was invoked by Install Manager or another JCL file containing a valid Job statement). It instructs Batch Manager to read the following:

- job name for the JCL file (up to 12 characters)
- user name for accessing user configuration data
- SysOut file name

Format: $Job jobname, [username, password, SysOut file]

where: **jobname** is the name used to reference the job and must be one to 12 characters in length.

username is the name (for example Darryl) used to access the user configuration file (for example, [Sys]<Sys>Darryl.User). Batch Manager looks locally for this file.

For batch processing in the foreground partition, the user name specified in the job statement replaces the signed-on user name once Batch execution is complete.

If you execute a Batch job in the background without specifying a user name, Batch opens the [Sys]<Sys>.user file or the [!Sys]<Sys>.user file if you are using a cluster workstation. If the user file does not exist, Batch returns an error.

Password is used to authorize access to protected areas.

SysOut file is the job output destination, usually a user-defined disk file. When specifying this file, include the complete path name, or the SysOut file will be created in the current path.

If a job is executed in the foreground and the SysOut file is not defined, Batch writes to the screen, and no SysOut file is created. If this statement is executed in the background and the SysOut file is not defined, Batch creates a default file, **[Scr]<Batch>Jobname.sysout$$Date-Time.log**. For Batch to create this file, the **<Batch>** directory must already exist and must not be password-protected.

Examples: $Job Budget,AcctDept,AcctDeptPswd ,[anotherSplQ]

$Job Compile,Bob,BobsPswd,[Sys]<Sys> Diskfile.SysOut

$Job Example,Bob,,[SPLB] (specifies SysOut file as a printer spooler file queued in the SPLB printer spooler queue)

$Job Example,Bob,,[Sys]<Sys>Example.Out (specifies SysOut file as the disk file [Sys]<Sys>Example.Out)

$Job Example,Bob (If this is a foreground job, no SysOut file is created. If this is a background job, then the default SysOut file is created.)

Log

The **Log** statement instructs Batch Manager to record a message in the Batch log file ([Scr]<Batch>Batch00.log, Batch01.log...). This statement is skipped in a foreground batch job.

Log differs from **LogStatus** in that **Log** writes to the Batch log file and **LogStatus** writes to Log.sys, which can be viewed using the Executive **PLog** command.

Format: $Log 'message'

where: message is any ASCII string, enclosed in single quotation marks.

Example: $Log 'Compiling and linking budget analysis program'

LogStatus

This statement instructs Batch Manager to place an entry in the system error log (**[Sys]<Sys>Log.sys**).

Note: *For **LogStatus** to function, you must specify a log file size when you initialize the volume containing the log file.*

LogStatus differs from **Log**. **LogStatus** is available in both foreground and background and writes to **[Sys]<Sys>Log.sys**, which can be viewed using the Executive command **PLog**. **Log**, however, is available only in background, and writes to the Batch log file (**[Scr]<Batch>Batchnn.log**).

Format: $LogStatus (value...)

where: value is a string literal or a string variable to be written to the [Sys]<Sys>Log.sys file;

value has a combined maximum length of 70 characters in real mode and 124 characters in protected mode. Any additional characters are truncated.

Do not include the tab character in the string text.

Example: $LogStatus ('Completed weekly backup')

Result: **Completed weekly backup** is written to **[Sys]<Sys>Log.sys**.

Example: $LogStatus('Erc = ',erc)

Result: **'Erc = '** followed by the appropriate status code is written to **[Sys]<Sys>Log.sys**.

NextFloppy

This statement, for foreground only, prompts you to insert a diskette with a given name. Batch waits until you have completed this action.

Format: $NextFloppy(FloppyName)

where: FloppyName is a string literal or a string variable.

Example 1: $NextFloppy('Std Sw 1 of 1')

Example 2:
```
$FloppyName = 'Std Sw 1 of 1'
$NextFloppy(FloppyName)
```

Result: **Insert Std Sw 1 of 1.**

(Press GO to continue, CANCEL, or FINISH to exit)

Batch tries to open "**[f0]<Sys>Std Sw 1 of 1**" until it succeeds or you press the **FINISH** key. You must create a file on the diskette named "**[f0]<Sys>Std Sw 1 of 1**"; it can be of zero length.

NumToStr()

This statement converts a number to a string and assigns the string to a string variable.

This function is useful when you must concatenate a numeric value onto a string variable. It can be parsed out later using the **Substring** and **StringToNum** statements.

Note: *Batch numeric variables are internally represented as DWord. The maximum value you can assign to a numeric variable is 4,294,967,295.*

Format: sTemp = NumToStr(number)
sTemp is the string variable;

number is a numeric constant, an expression, or a batch variable.

Example: $Assign nvar,786

sTemp = NumToStr(nvar)

same as: $Assign sTemp,'786'

Path

This statement instructs Batch Manager to set the operating system's file access path. This JCL **Path** statement is preferable to the Executive **Path** command because it is internal to batch processing and is therefore faster. (Refer to the *CTOS Executive Reference Manual* for more information.)

Format: $Path node, vol, dir, prefix, password

where: node is the node name;

vol is the volume name;

dir is the directory name;

prefix is the file prefix;

password is the user password.

Example 1: $Path ,Win1, Bob, Account, MyPassword ; changes everything except the node

Example 2: $Path , , Bob ; only changes directory

Example 3: $Path , , , Test ; only changes prefix

Example 4: $Path {local}, , Build ; changes node and directory

Note: *When the **Path** statement is executed in the background, Batch will not change the path that appears on the Executive display.*

PauseOff

This statement is one of the video commands that allows Batch Manager foreground contexts to take advantage of system video characteristics. **PauseOff** causes the system to scroll information on the screen without pausing. (Also refer to **PauseOn**.)

Refer to the *CTOS Operating System Concepts Manual* for more information on system video attributes.

Format: $PauseOff

PauseOn

This statement is one of the video commands that allows Batch Manager foreground contexts to take advantage of system video characteristics. **PauseOn** causes the system to pause and prompt you to press **NEXT PAGE** before scrolling information off the screen. (Also refer to **PauseOff**.)

Refer the *CTOS Operating System Concepts Manual* for more information on system video attributes.

Format: $PauseOn

Prefix

This statement instructs Batch Manager to set an other-than-default prefix for an implicit call construct. If the **Prefix** statement is not used, the default prefix is **[Sys]<Sys>**. (Refer to **Implicit Call** for more information.)

Format: $Prefix {node}[vol]<dir>

where: node is the node name (optional);

vol is the volume name;

dir is the directory name.

Example 1: $Prefix {NY}[Win]<Bob>

Example 2: $Prefix [D2]<Build>

Reboot

This statement causes the workstation to reboot. For example, it is useful after installing software that would need to reboot to access the new request codes.

Reboot acts just like pressing the Reset button on your system.

Format: $Reboot

Caution:

Any adverse effects from the use of this statement are the responsibility of the JCL writer.

RestartLabel

This statement gives Batch Manager a starting point when a JCL file terminates abnormally. It should be inserted wherever you want the JCL file to resume if Batch Manager terminates before the file is completely processed.

When a **Yes** value is entered for the **Restart JCL File** parameter of the **Batch Foreground** command, Batch Manager recalls the last **RestartLabel** encountered and starts execution from that point. **RestartLabel** functions only in the foreground.

The **RestartLabel** entries must be typed into the JCL file; for example, if a JCL file copies files and then creates commands, a logical restart point is at the command creation. Failure at that point, and restarting, allows Batch to skip the copy.

Note: *Restarting is of little use during system initialization. The Sysinit.jcl file usually installs system services, and in the event of a crash, the JCL file needs to be processed from the beginning.*

Attempting to use **RestartLabel** in a file that resides on a write-protected diskette will result in an error. This is because **RestartLabel** writes information to the file header. First, copy the file from the floppy diskette to a hard disk, then access the file from the hard disk.

Format: $RestartLabel

Example:
```
$RestartLabel
$Run [sys]<sys>Lcopy.run, NewProgram,
[F0]<Sys>, [D1]<New>,.run
..
;create new commands
$RestartLabel
; <crashes here>...
```

Result: Using the **Batch Foreground** or **Batch command,** you can enter a **Yes** value in the **[Restart JCL File?]** field.

Batch Manager then resumes execution in this JCL file from the **RestartLabel** listed prior to the crash comment. The **LCopy** statement and everything above the second **RestartLabel** is bypassed.

Return

This statement instructs Batch Manager to return control to the calling JCL file from the called JCL file. Any JCL file started by a **Call** statement or by an implicit call construct can use **Return** to terminate execution and return to its parent JCL file.

If **Return** is used in the top level JCL file, it acts like an **End** statement and returns control to the Executive.

Format: $Return

Run

This statement instructs Batch Manager to load and activate a run file and pass program parameters to a run file. When the run statement is invoked, Batch sets the Exit run file to batch.run in foreground Batch or BathMgr.run in background and then chains to the run file. When the specified run file has finished execution, control returns to Batch Manager.

Since control returns to Batch through the Exit run file, the run file specified must not change the Exit run file. Otherwise, Batch will not resume execution when the run file finishes execution.

When executing in background, you must supply the yes/no parameters in your JCL file for commands which require user confirmation (for example, **Copy** or **Rename**).

Format 1: $Run (runfilespec[, command name, case value])
[, parameters]

Format 2: $Run runfilespec[, parameters]

Format 3: $Run runfilespec(subparam1, subparam2), param2

where: **runfilespec** is the full file specification;

command name is a command to be passed as a parameter;

case value is the command case to be passed to the run file;

parameters are program parameters to be passed to the run file.

Example 1:
```
$Job BatchJob1, Randy, Secret, B1.out
$Run ([Sys]<Sys>InstallMgr.run, 'Floppy Install', 'IF'), y
```

Example 2:
```
$Run {NY}[Win]<Bob>FileA.Run, ParameterA
```

Example 3: $Run ([Sys]<Sys>COBOL.Run, COBOL), ProgramA

Example 4: $Run [sys]<sys>CC0.run,
(-mlf,-I[sys]<BTOSC>),Program.c

Program parameters may include the wild card characters question mark and asterisk (? and *), and the at-sign (@) file character.

Example 5: $Run [Sys]<Sys>SelectiveBackup.Run,@Pascal.fls

If a runfile expression is not expanded by the Executive, and you want the run file (that you are trying to execute) to expand the wild cards, then the statement must be enclosed in single quotes. However, if you want Batch to do the expanding, leave the quotes off. Batch expansion of wildcards is equivalent to wild card expansion in the Executive.

Examples: $Run Files.run, '[sys]<sys>*.user'

$Run Files.run, [sys]<sys>*.user

In the first example, the wildcard is expanded by Files.run. In the second example, the wildcard is expanded by Batch Manager.

You cannot specify [Sys]<Sys>Exec.run in a **Run** statement. If you include [Sys]<Sys>CmdFileEditor.run in a **Run** statement, you must specify the command case.

Unisys recommends using **$Command** instead of **$Run** whenever possible for portability across domestic systems. With **$Run,** Batch Manager looks for the command only in the directory that is explicitly specified. If your System Administrator reorganizes directories or if you attempt to execute your JCL files on another system, your **$Run** statements will fail because the commands invoked will be in different directories. On the other hand, **$Command** statements will continue to work regardless of the locations of the commands invoked.

*Note: To run another JCL file, you must use **$Call,** not **$Run [Sys]<Sys>Batch.run. $Run [Sys]<Sys>Batch.run** causes Batch Manager to call itself and keep running, never executing the desired JCL file.*

RunNoWait

This statement, specifically for the XE shared resource processor, causes Batch on the server processor, to continue processing without waiting for a termination response from the board executing the **RunNoWait** statement.

You can refer to Appendix C, System Initialization Examples, for a more detailed example of the **RunNoWait** statement.

Format: $RunNoWait (runfilespec[, command name, case value]) [,parameters]

or

$RunNoWait runfilespec[, parameters]

or

$RunNoWait runfilespec(subparam1, subparam2), param2

Examples:
```
CP00
$RunNoWait ([NY][Win]<Bob>FileA.Run), ParameterA

FP00
$RunNoWait [Sys]<Sys>SelectiveBackup.Run, '\@Pascal.fls'
```

SendEvent

This statement generates an event for Event Manager. You use this statement when sending an event other than those automatically generated by Batch Manager.

The **ReturnPt, Class, Topic, Activity, PenId,** and **ContextMessage** parameters are equivalent to the Local Event Management **SendEvent** parameters. For further explanation of these **SendEvent** parameters, refer to the *CTOS Local Event Management Event Manager Administration and Programming Guide.*

Format: $SendEvent ReturnPt, Class, Topic, Activity, PenId, ContextMessage, Erc, SubTopic, MsgType, Msg, MsgType, XtraMsg

where: **ReturnPt** is a numeric value that specifies the point at which Batch should return from procedural interface contact with Event Manager.

Class is a numeric value that describes the relative severity of the condition or problem defined in an event.

Topic is a numeric value that identifies the general area affected by an event or its consequences.

Activity identifies the incident or action that caused the event.

PenId identifies the software component (program or system) that detects the occurence of an event and sends an event message to Event Manager.

ContextMessage, Msg, and **XtraMsg** are string variables you use to pass messages that correspond to events. Message text stored in **ContextMessage, Msg,** and **XtraMsg** should total no more than 1024 bytes in length.

Erc represents a status code number.

SubTopic is a numeric value used together with **Topic** to identify the area affected by an event.

MsgType identifies the type of data contained in the **Msg,** and **XtraMsg** fields. Valid types are:

2 string
4 unsigned integer

SendMail

This statement enables you to send a mail message to a specific user or to all users in the message distribution list. Your .user file must contain the keyword for your OFIS Mail user ID (for example, :MailUserName:J.Doe). OFIS Mail must not be running in another partition when this statement runs.

For an attachment, you must specify a full qualified file name. The mail message will be sent from the user specified in the **$Job** card.

Format: $SendMail (username, message[, subject][, attachment])

where: subject and attachment are optional.

Examples: $SendMail ('T. Dooley', 'Test Message')

$SendMail ('P. Martini', 'Test Message', 'Testing')

$Assign Susan, 'S. Smith'

$SendMail (Susan, 'Test Message', 'Testing Only', '[Sys]<Sys>File1')

$SendMail ('!TestGroup', 'Test Message')

StringLength()

This statement assigns the length of an existing string variable to a numeric variable. The string length of a null string is always 0.

Example:
```
text = 'A long string to test'
chars = StringLength (text)
```

Result: chars = 21

StrToNum()

This statement is used to convert a numeric string to a number and assign the number to a numeric variable.

This statement is useful in conjunction with **CopyString** and **StringLength** statements.

Note: *Batch numeric variables are internally represented as DWord.*

Format: VariableName = StrToNum(numstring)

where: VariableName is a numeric batch variable;

numstring is a numeric string (enclosed in single quotation marks inside the parentheses) or a Batch variable to which a numeric string has been assigned.

Example:
```
NumString = '65535'
var3 = StrToNum(numstring)
```

or

```
var3 = StrToNum ('65535')
var4 = var3 - 5535
```

Result: Var3 = 60000

SubString()

This statement instructs Batch Manager to search one string for another string. The index of the first occurrence of the matching string is returned. Indexing starts at 0. If no match occurs, the value 65535 is returned. Case sensitivity is ignored.

This statement is useful in conjunction with **CopyString** and **StringLength** statements.

Format: nVar = SubString(str1, str2)

where: nVar is the numeric variable to which the index or 65535 is returned;

str1 is the string to be sought;

str2 is the string to be searched.

Example 1: $Assign str1, 'Test'
$Assign str2, 'This is a test string'
nVar = Substring(str1, str2)

Result: nVar = 10

Example 2: $Assign str1, 'Test'
$Assign str2, 'Retest the test string'
nVar = Substring(str1, str2)

Result: nVar = 2

since test is first found in the word 'Retest'.

Suffix

This statement instructs Batch Manager to set an other-than-default suffix for an implicit call construct. If the **Suffix** statement is not used, **.jcl** is the default suffix. (Refer to **Implicit Call** for more information.)

Format: $Suffix suffix

where: suffix is the suffix set for an implicit call construct.

Example:
```
$Suffix .Batchjcl
$Test
```

Result: [Sys]<Sys>Test.Batchjcl is called

Example:
```
$Suffix Myjcl
$Test
```

Result: [Sys]<Sys>TestMyjcl is called

UserEnterValue()

This statement displays a prompt and a box into which you can make an entry. It returns a string if you made an entry, null or the default if you did not. This statement will not execute in background operation. Refer to Figure 6-1 for the display of both examples below.

The combined length of prompt and default or user entry must fit on one line within the pop-up window; the maximum length is 72 characters.

Format: Var = UserEnterValue(prompt [,default])

where: **default** is an optional string variable.

The maximum length of default/entry is 30 characters.

Example 1:
```
Val = 'Enter a value'
Values = '10000'

Choice = UserEnterValue(Val,Values)
```

The upper pop-up box in Figure 6-1 illustrates the use of **UserEnterValue** using the parameters in Example 1.

Example 2: If the entered value needs to be protected (such as a password), add a caret (^) as the last character in the prompt. The entered value displays as hash marks (####).

```
x = UserEnterValue ('Enter a password^')
```

or

```
Prompt = 'Enter a password ^'
x = UserEnterValue (prompt)
```

The lower pop-up box in figure 6-1 illustrates the use of **UserEnterValue** using the parameters in Example 2.

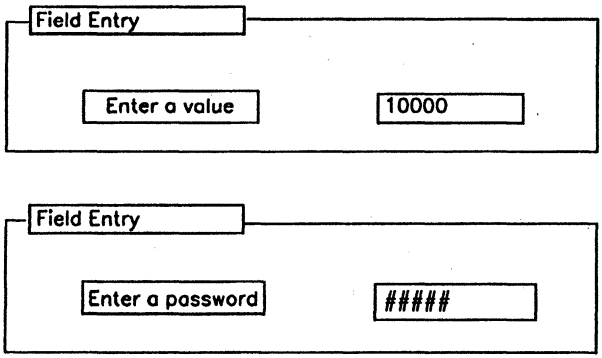

Figure 6-1. UserEnterValue displays

UserSelectMultiple()

This statement displays a prompt and a menu of items. If you select more than one item, a list of the selected items is returned. If you select one item only, the result is exactly that of **UserSelectSingle**. This statement will not execute in a background partition. See Figure 6-2 for the display.

The maximum number of items to display is 12, with each item having a maximum length of 72 characters.

Format: x = UserSelectMultiple(prompt, item0, item1, item2, ...)

Example:
```
Prompt = 'Make a selection.'
TestA = 'TestsetA'
TestB = 'TestsetB'
TestC = 'TestsetC'
NotFound = ''
Choice = UserSelectMultiple(Prompt, TestA, TestB, TestC)
If (Choice = NotFound)
.
.
.
```

In Figure 6-2, the display indicates that you have selected both TestSetA and TestSetB.

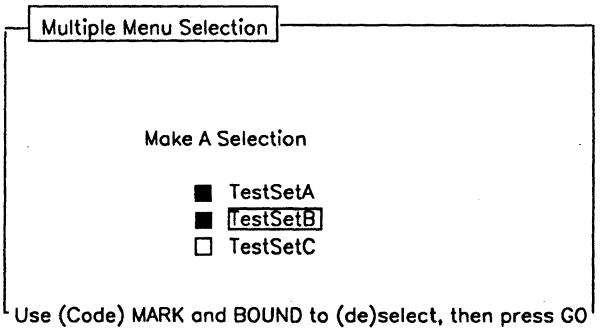

Figure 6-2. UserSelectMultiple display

UserSelectSingle()

This statement displays a prompt and a menu of items. If you select an item, the string value of that item is returned. This statement will not execute in background operation. See Figure 6-3 for the display.

The maximum number of items to display is 12, with each item having a maximum length of 72 characters.

Format: x = UserSelectSingle(prompt, item0, item1, item2, ...)

Example:
```
Prompt = 'Make A Selection.'
TestA = 'TestsetA'
TestB = 'TestsetB'
TestC = 'TestsetC'
Choice = UserSelectSingle(Prompt, TestA, TestB, TestC)
If Choice = TestA
.
.
.
```

In Figure 6-3, the display indicates that TestsetA has been selected.

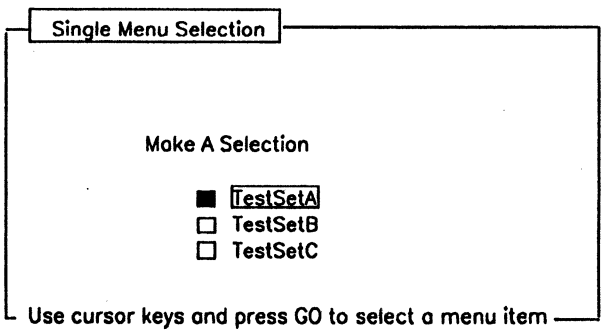

Figure 6-3. UserSelectSingle display

UserSelectYesNo()

This statement displays a prompt and a box into which you can make a selection. It returns 255 if the response is **Yes**; 0 if the response is **No**. This statement will not execute in background operation. See Figure 6-4 for the display.

The window can display a maximum width of 72 characters.

Format: x = UserSelectYesNo(prompt, [default])

where: default is optional.

Example: Prompt = 'Select Yes or No'
Default = 'No'
Choice = UserSelectYesNo(Prompt, Default)

In Figure 6-4, the display indicates that the **No** option has been selected.

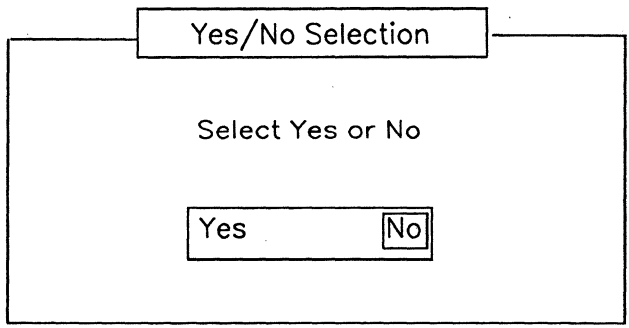

Figure 6-4. UserSelectYesNo display

While

This statement instructs Batch Manager to continue looping within the **While-EndWhile** loop until the conditional expression of the **While** statment is no longer true. There must be an **EndWhile** statement for each **While** statement.

The maximum number of statement that can be nested is 12.

Format:

```
$While (conditional expression)

.
.
.
$EndWhile
```

Example:

```
$EchoOff
x = 5
$While x LE 10
   $DisplayLine (x)
   x = x + 1
$EndWhile
```

Result:

```
5
6
7
8
9
10
```

Appendix A
Troubleshooting

This appendix contains information on various commonly occurring Batch Manager problems and solutions for these problems. The information provided is divided into an alphabetical listing of status messages, and a topical listing of commonly occurring problems.

Status Messages

This subsection contains a partial listing of Batch status messages in alphabetical order. If a status message indicates an entry error, the entry describes the probable cause and suggests corrective action.

*****Warning*** - negative number conversion**

You can use negative numeric values in mathematical equations; however, Batch cannot display a negative number.

A Batch Manager cannot be installed on a workstation executing a compact OS

The system has booted off a diskette. Use a regular operating system.

Bad expression

There is a syntax error in the current expression.

Bad file specification

Check for full file specifications or typographical error in the [Vol]<Dir>FileName format.

Bad memory size parameter

An invalid memory size was used. Correct and retry.

Bad priority parameter

An incorrect job priority was entered. Check the range for the parameter and enter an appropriate number.

Bad time parameter

An incorrect date or time was entered for the specified parameter. Examine the entry and correct the error.

Bad yes/no parameter

The specified yes or no parameter is invalid. Examine the entry and correct the error.

Batch foreground is an invalid command in a JCL file

To execute another JCL file from within a JCL file, use Call or Implicit Call constructs.

Batch Manager installation failed. Queue table full.

Allocate more queues when installing Queue Manager.

Cannot access the [!Scr]<Batch> directory, error =

An attempt was made to access a nonexistent directory. Create the directory and retry.

Cannot access user profile file in [Sys]<Sys> directory

The user name and password combination given in the Job statement is incorrect. Examine the entry and correct the error.

Cannot find :MailUserName: entry in user file

The user specified in the BroadcastMessage or SendMail statement cannot be found. Check the user file to ensure :MailUserName: token is specified.

Cannot open specified JCL file

An incorrect or unrecognized JCL file name was entered. Recreate the file or correct the entry and rerun the job. This also displays when Batch attempts to recognize a nonexistent implicit call when, in reality, you probably made a typographical error on a keyword.

Cannot re-open the Batch Context file

A problem occurred in foreground batch. Follow the instructions for recovery of foreground jobs; you may need to begin again.

Cannot set path with data in user profile file

The .user file requires entries for a specific path (node, volume, and directory).

Chaining error

Chaining to the run file Batch.run has failed due to the returned error number (refer to your Status Codes Reference Manual for further information).

Duplicate labels

Batch cannot maintain two labels with the same name in the same JCL file. Change the name of one of the labels.

GoTo nonexistent label

A GoTo statement cannot find the specified label. Make sure the label exists and its name is specified.

Illegal string operation

An invalid string operation was made. Ensure only EQ (=) and NE (<>) are used.

Insufficient memory available to create the specified Batch partition

The workstation does not have enough memory to run Batch Manager. If running under Context Manager, increase the partition size.

Invalid variable name

The variable name does not follow the Batch Manager variable naming syntax. Check to ensure that it starts with a letter rather than a number and that it has been assigned.

Nesting too deep

The nesting level of the current If or While statement is more than 12. Reorganize the statements so that the nesting does not exceed 12 levels.

No available control queues, edit [Sys]<Sys>Queue.index

There are no control queues specified for the currently running job. Edit the Queue.Index file to make entries for the control queues.

No Batch partitions are active

Install Batch to create an active Batch queue.

No Batch Supervisor currently executing

Install Batch to create a Batch supervisor and Batch Manager.

No such keyword

This JCL statement cannot be found. Check for typographical errors.

Non numeric value

The operation which is trying to be performed requires a numeric value, not a string.

Partition size too small

The minimum size for a batch partition is 300K. Make the partition size parameter in the form equal to or greater than 300K.

Processor not found

The configuration does not include the specified board (for example, FP00 or CP00).

Quotes are unbalanced

A subparameter containing an embedded space was not enclosed with single quotation marks. Correct the entry and reenter the command.

Remote processor is not accepting commands

Processor boards, on a shared resource processor, are not accepting JCL instructions from the server board.

String too long

The string causing the error is longer than the maximum of 2K.

Symbol table is full

A maximum of 64 local variables per JCL file and 64 global variables overall can be assigned. Deassign some unneeded variables using Deassign or DeassignLocal statements.

SysIn/SysOut bytestream is not installed.

Install the bytestream service before installing Batch.

Syntax error on current command line

Batch cannot process the current JCL statement. Verify that its syntax is correct.

The JCL file does not contain a valid $JOB statement

All or part of the specified Job statement is missing. Examine the entry and correct the error.

The queue manager has not been installed

The Queue Manager is not installed. Use the Install Queue Manager command to install Queue Manager, and submit the job again.

The specified batch queue does not exist

An invalid Batch Queue name was entered. Use the Batch Status command to find the correct name and enter it.

This file specification is illegal

Verify that the syntax of the file specification is correct; that is, {Node}[Vol]<Dir>FileName.

This variable is a read only variable

An attempt was made to write to a read only variable.

Too deeply nested

A JCL file is calling too many other files. Maximum number of nested files is 12. Reduce the number of CALL statements.

Too many secondary partitions

No more secondary partitions can be allocated because the maximum number of partitions have already been allocated.

Type conflict

The variables in question are not of the same data type. Check their declaration to ensure that they refer to the same data type.

Unable to access Batch Status queue

Either the Batch Status Queue entry in the [Sys]<Sys>Queue.Index file is missing or it is invalid. Edit the Queue.Index file to correct the entry.

Unable to chain to [Sys]<Sys>BatchMgr.run

This file may not exist, may reside in a different path, may already be in use, or not enough memory is available to run it.

Unbalanced quotes in the Parameters field

A quote was left off in this field. Enclose the affected parameter in single quote marks.

Undeclared variable:

Batch does not recognize this variable. Make sure that it is defined and is spelled correctly.

Unexpected Else

An Else statement was used without an If statement. Edit the file to insert a matching If statement.

Unrecognized command

The command specified in the batch file is not a valid command. Edit the batch file and correct the command.

Unrecognized EndIf

An EndIf statement was encountered where there was no corresponding If statement. Edit the file to insert a matching If statement.

Unrecognized EndWhile

An EndWhile statement was used without using a While statement before it. Edit the file to insert a matching While statement.

Variable name too long

The variable name in question is more than 12 characters in length. Shorten it so that it is 12 characters or less.

Common Batch Problems

This subsection contains a list of commonly encountered problems and corresponding solutions. Problems included address:

- Software installation
- System initialization on an SRP
- Memory limitations
- File corruption
- Reading log files during background execution
- Misspelling names of JCL statements

This section also provides information on debugging Batch files.

Software Installation Problems

Problem: Batch installation failed using Floppy Install and an old version of Batch.run exists on the system.

Solution: Copy **Batch.run** and **Batchmsg.bin** from the installation diskette to [Sys]<Sys> and reinstall Batch.

Problem: Attempts to install Batch software using Installation Manager return a file not found (Erc 203) for the file **[Sys]<sys>Install>English.Cmds**.

Solution: Copy **Install>English.Cmds** from the server to [Sys]<Sys>, and reinstall the Batch software.

System Initialization on a Shared Resource Processor

Problem: System initialization on a shared resource processor results in the display of a tilde (~) on a real mode processor when using Admin Clusterview. This is caused by a difference in processing speed between the server and cluster processors.

Solution: To begin normal operation of the cluster workstation, press **ACTION-FINISH**. Processing resumes, and the workstation invokes the exit run file SignOn.run.

Memory Problems

Problem: Error 400 is returned in Batch background SysOut file.

Solutions:

- Remove the Batch partition and reinstall the partition with more memory.
- Check to see whether you attempted to install Window Services before installing Batch. If so, install the application before Window Services.

Problem: Install Batch failed and the batch partition was not created.

Solution: Use the Executive command "**Plog**" and look for error messages. Batch writes error messages to log.sys.

Problem: A Batch partition cannot be removed using the remove key in Batch Status.

Solution: If the run file exists on your system, run the file Removepartition.run and enter the Batch partition as a parameter. For example:

```
Run
  Run file        [Sys]<sys>RemovePartition.run
  [Case]
  [Command]
  [Parameter 1]   batchpart00
  [Parameter 2]
  [Parameter 3]
```

File Corruption

Problem: Information in the context files becomes corrupt.

Solution: Use the following procedure to delete the corrupted files.

1. Remove the queue, using the Batch Status Remove key or RemovePartition.run.
2. Deinstall the Queue Manager
3. Delete the following files:
   - Batch queue files from the Batch directory
   - Batch context file from the $ directory
   - Batch related files from the SPL directory
4. Reinstall Queue Manager
5. Reinstall the Batch partition.

Reading Batch Log Files During Background Execution

Problem: You wish to consult the log file while Batch is executing jobs in the background without terminating jobs; however, information in the Queuenamenn.log file is unavailable while Queue Manager and Batch Manager are installed.

Solution: To read the contents of the log file without deinstalling the Queue Manager or Batch Manager, you use the Batch Status utility to copy the contents of the log file to a temporary file and then access the temporary file through the CTOS Editor.

1. Invoke the **Batch Status** utility.
2. Press **Print (F3)** from the Batch Status Main display to copy the **Queuenamenn.log** file to a .tmp file in the spooler directory.
3. Log out of the **Batch Status** utility.
4. Use an Executive **Files** command to find the name of the temporary file in your spooler directory.

   The file name appears in the format <spl>QueuenameLog$$time.tmp (for example <spl>BatchLog$$15:22:39.tmp).
5. Use the Executive **Type** or **Editor** command to access the contents of the temporary file.

Misspelling Names of JCL Statements

Problem: Batch Manager returns error 203 when you attempt to execute a JCL statement.

Solution: You misspelled the name of the JCL statement, and Batch Manager attempted to carry out an implicit call. Correct the spelling and run the batch job again.

Debugging JCL Files

When writing JCL files, you can use the **Dump** statement and **DumpValue** variable for debugging purposes.

If you are debugging a JCL file, you may want to use **RestartLabel** statements to speed execution. If you enter a **RestartLabel** statement in the JCL file and the execution of the JCL file failed for some reason, correct the error and execute the job again, entering **Yes** in the **Restart JCL File** parameter. Execution restarts at the last processed **RestartLabel** statement. Without a **RestartLabel** statement, execution starts at the beginning of the file.

Appendix B
Creating Batch Queues

You can take advantage of Queue Manager dynamic queue creation, or, you may bypass this dynamic allocation by specifying your own queue configuration in the queue.index file (examples are provided later in this appendix).

Batch Manager requires three types of queues for background batch processing:

- the Batch Scheduling queue (one or more for each cluster configuration)
- the Batch Control queue (one for each Batch Manager)
- the Batch Status queue (one for each cluster configuration)

Each statement in the following examples contains queue information in the form **/X/Y** where:

X is the entry size of the the queue entry file in sectors (512 bytes each)

Y is the queue type (type is 3 for all Batch queues).

For more information, see Batch Scheduling Queue in this appendix.

Before installing Queue Manager on your workstation (refer to the *CTOS Executive Reference Manual* for installation procedures), you can edit the file **[Sys]<Sys>Queue.Index** to change the necessary batch queues.

Example-1 shows the contents of the Queue.Index file which has been edited for one Batch Manager. This file includes:

one Batch Scheduling Queue, named JOE (identified as number 1)

one Batch Control Queue (identified as number 2)

one Batch Status Queue (identified as number 3)

```
   SPL/[Sys]<Spl>SPL.QUEUE/1/1
   PARALLELCONTROL/[Sys]<Spl>PARALLELCONTROL.QUEUE/1/1
   SERIALCONTROL/[Sys]<Spl>SERIALCONTROL.QUEUE/1/1
   PARALLELCONTROL/[Sys]<Spl>PARALLELCONTROL.QUEUE/1/1
   SERIALCONTROL/[Sys]<Spl>SERIALCONTROL.QUEUE/1/1
   SPLB/[Sys]<Spl>SPLB.QUEUE/1/1
   SPOOLERSTATUS/[Sys]<Spl>SPOOLERSTATUS.QUEUE/1/1
1  JOE/[Sys]<Batch>JOE.QUEUE/1/3
2  BATCHCONTROL00/[Sys]<Batch>BATCHCONTROL00.QUEUE/1/3
3  BATCHSTATUS/[Sys]<Batch>BATCHSTATUS.QUEUE/1/3
```

Example1. Queue.Index File Edited for One Batch Manager

Example-2 shows a Queue.Index file that has been edited for two Batch Managers:

one Batch Scheduling Queue, named JOE (identified as number 1)

two Batch Control Queues (identified as number 2)

one Batch Status Queue (identified as number 3)

```
   SPL/[Sys]<Spl>SPL.QUEUE/1/1
   PARALLELCONTROL/[Sys]<Spl>PARACONTROL.QUEUE/1/1
   SERIALCONTROL/[Sys]<Spl>SERIALCONTROL.QUEUE/1/1
   SPLB/[Sys]<Spl>SPLB.QUEUE/1/1
   SPOOLERSTATUS/[Sys]<Spl>SPOOLERSTATUS.QUEUE/1/1
1  JOE/[Sys]<Batch>JOE.QUEUE/1/3
2  BATCHCONTROL00/[Sys]<Batch>BATCHCONTROL00.QUEUE/1/3
2  BATCHCONTROL01/[Sys]<Batch>BATCHCONTROL01.QUEUE/1/3
3  BATCHSTATUS/[Sys]<Batch>BATCHSTATUS.QUEUE/1/3
```

Example 2. Queue.Index File Edited for Two Batch Managers and One Batch Scheduling Queue

Example-3 shows a Queue.Index file that has been edited for two Batch Managers and two Batch scheduling queues:

two Batch Scheduling Queues, named JOE and ANN (identified as number 1)

two Batch Control Queues (identified as number 2)

one Batch Status Queue (identified as number 3)

```
    SPL/[Sys]<Spl>SPL.QUEUE/1/1
    PARALLELCONTROL/[Sys]<Spl>PARACONTROL.QUEUE/1/1
    SERIALCONTROL/[Sys]<Spl>SERIALCONTROL.QUEUE/1/1
    SPLB/[Sys]<Spl>SPLB.QUEUE/1/1
    SPOOLERSTATUS/[Sys]<Spl>SPOOLERSTATUS.QUEUE/1/1
1   JOE/[Sys]<Batch>JOE.QUEUE/1/3
1   ANN/[Sys]<Batch>ANN.QUEUE/1/3
2   BATCHCONTROL00/[Sys]<Batch>BATCHCONTROL00.QUEUE/1/3
2   BATCHCONTROL01/[Sys]<Batch>BATCHCONTROL01.QUEUE/1/3
3   BATCHSTATUS/[Sys]<Batch>BATCHSTATUS.QUEUE/1/3
```

Example 3. Queue.Index File Edited for Two Batch Managers and Two Batch Scheduling Queues

In the queue.index files shown in example 4, **Batch** is the Batch queue on {node1} and **Batch1** is the Batch queue on {node2}.

Queue.Index file for BNet {Node1}:

```
SPL/[Sys]<Spl>SPL.QUEUE/1/1
CENTRONIXCONTROL/[Sys]<Spl>CENTRONIXCONTROL.QUEUE/1/1
DIABLOCONTROL/[Sys]<Spl>DIABLOCONTROL.QUEUE/1/1
SPLB/[Sys]<Spl>SPLB.QUEUE/1/1
SPOOLERSTATUS/[Sys]<Spl>SPOOLERSTATUS.QUEUE/1/1
BATCH/[sys]<batch>BATCH.QUEUE/1/3
BATCHCONTROL00/[sys]<batch>BATCHCONTROL00.QUEUE/1/3
BATCHSTATUS/[sys]<batch>BATCHSTATUS.QUEUE/1/3
{node2}BATCH1/[sys]<batch>BATCH1.QUEUE/1/3
```

Queue.Index file for BNet {Node2}:

```
SPL/[Sys]<Spl>SPL.QUEUE/1/1
CENTRONIXCONTROL/[Sys]<Spl>CENTRONIXCONTROL.QUEUE/1/1
DIABLOCONTROL/[Sys]<Spl>DIABLOCONTROL.QUEUE/1/1
SPLB/[Sys]<Spl>SPLB.QUEUE/1/1
SPOOLERSTATUS/[Sys]<Spl>SPOOLERSTATUS.QUEUE/1/1
BATCH1/[sys]<batch>BATCH1.QUEUE/1/3
BATCHCONTROL00/[sys]<batch>BATCHCONTROL00.QUEUE/1/3
BATCHSTATUS/[sys]<batch>BATCHSTATUS.QUEUE/1/3
```

BNet sample JCL file:

```
$job bnet,steve,,[sys]<sys>bnet.log
$Command batch, [sys]<sys>node1.jcl,,{node2}[batch1]
$run [sys]<sys>batch.run,{node2}[sys]<sys>node2.jcl,,{node2}[batch1]
$end
```

Example 4. Queue.Index Files and Sample .JCL file for BNet Nodes BNet

The corresponding sample .jcl file BNet.jcl shown here is processed on {node1}. **node1.jcl** and **node2.jcl** are .jcl files.

The first statement in the BNet sample JCL file runs **node1.jcl** on the **Batch1** queue of **{node2}**. The second statement in BNet-example.jcl runs **node2.jcl** on the **Batch1** queue of **{node2}**.

When the JCL file is located at another node or if the **Path** statement is used within the JCL file to access another node, you must provide the full JCL file specification (node, volume, and directory). To use the **Batch** command to run a job in which both the JCL file and the Batch queue are located at a remote node, enter:

```
Batch
  JCL File                          {nodename}[Sys]<Sys>JOB1.JCL
  [Parameters]
  [Batch queue]                     {nodename}[BATCH1]
  [After date/time]
  [Priority]
  [Repeat After Time]
  [Job Expiration Date & Time]
  [User role (for Access Control)]
```

Batch Scheduling Queue

Each Batch Scheduling Queue contains entries describing the jobs to be processed. A queue entry contains the JCL file name, job name, time the job was submitted, name of user submitting the job, all parameters passed to the job, and the two-digit identification number of the Batch Manager processing the job (refer to Table B-1).

A Batch Manager serves the Batch Scheduling Queue specified in the second parameter of the **Install Batch** command.

The format for a Batch Scheduling Queue is:

QUEUENAME/Full file spec/1/3

Example: Joe/[Sys]<Batch>Joe.Queue/1/3.

You can queue a background batch job using either of the following methods:

- From the Executive command line, use the **Batch** command.
- From within an application, use the operating system procedure call **AddQueueEntry.**

To determine the status of a job from within an application, you can poll the scheduling queue to find a matching queue entry using the operating system procedure call **ReadNextQueueEntry.** (Refer to your *CTOS Operating System Concepts Manual* for more information.) The entry in the scheduling queue is deleted when the job terminates unless a **Repeat After Time** was specified when the job was queued.

Table B-1 shows the queue entry format for the Batch Scheduling Queue.

Table B-1. Batch Scheduling Queue Entry Format

Offset (bytes)	Field Name	Size (bytes)	Description
0	sbJCLfilespec	92	file specification for the scheduled JCL file
92	sbJobName	13	job name (as appears in the Job statement)
105	timeSubmitted	4	time when the job was submitted, in date/time format
109	sbUsername	31	name of the user who submitted the job
140	bManagerNum	1	identification number of the Batch Manager processing the job
141	cParams	1	number of parameters passed to the job
142	rgsbParams	278	packed array of sb strings that are passed parameters
420	sbSubjNodeName	13	name of the subject node obtained from the local access-control kernel
433	sbRoleName	31	name of the user role for access-control
464	timeExpire	4	time when the job is removed from the queue
468	timeRepeat	4	repeat interval

Batch Control Queue

If you choose to bypass Batch Manager's dynamically allocated queues, you must define a Batch Control Queue for each active Batch Manager in a cluster.

Each Batch Manager automatically receives the next free Batch Control Queue for managing such internal operations as canceling jobs, removing a Batch partition, and printing a Batch Log File. The name of each Batch Control Queue is BatchControl, followed by two digits (00 to nn, where nn is the number of active Batch Managers in the cluster). An example is BatchControl04. The appended digits appear as the identification number in the Batch Control Queue (refer to table B-2).

Note: *The Batch Control Queue is not directly accessible by invoking the* ***Batch Status*** *command.*

Table B-2 presents a sample Batch Control Queue entry format and field definitions.

Table B-2. Batch Control Queue Entry Format

Offset (bytes)	Field Name	Size (bytes)	Description
0	bCommand	1	operative command, set by function key selection on the Batch Status Queue display: 1 = cancel current job 2 = reserved 3 = remove Batch partition 4 = print log file
1	sbJobName	13	job name for the JCL file currently processing; required for the cancel job function
14	JobStepIdentifier	4	identifier for each job step currently processing; required for the cancel job step function

Batch Status Queue

The Batch Status Queue contains detailed status information about all active Batch Managers in a cluster. This information includes the job each Batch Manager is processing, the job step currently executing, and the name of the user who submitted the job. You can display this information with the **Batch Status** command.

Each active Batch Manager marks a queue entry in the Batch Status Queue and repeatedly updates this entry to reflect the current state of the Batch Manager.

Table B-3 shows the format of a Batch Status Queue.

Table B-3. Batch Status Queue Entry Format

Offset (bytes)	Field Name	Size (bytes)	Description
0	sbBatchQueue	51	name of the Batch queue served by the Batch Manager
51	sbJobName	13	name of the job currently processing (if sbJobname(0)=0, Batch Manager is idle)
64	sbJobStep	133	job step currently executing
197	cbManagerIdNum	1	always 1; used with the ReadKeyedQueue-Entry operation
198	bManagerNum	1	identification number of the Batch Manager processing the job
199	timeStarted	4	time the current job was started, in date/time format
203	sbUserName	31	name of the user who submitted the job
234	RESERVED	4	
238	bBatchStatus	1	Batch status 0 if executing, 0FFh if idle

Appendix C
System Initialization Examples

The SysInit.jcl file examples in this section use the **FrontPanel**, **ContinueOnError**, **Run**, and **EndBoard** statements. The board name label (for example, GP00) designates the board on which various services run.

RunNoWait, **CallNoWait**, board name labels (xPxx), and **EndBoard** are available for use only on the shared resource processor at system boot time.

The following JCL keywords will run on the subordinate boards designated by the board name labels:

- **$Command**
- **$Run**
- **$RunNoWait**
- **$Call**
- **$CallNoWait**
- **$End**

All other keywords will be executed by the shared resource processor GP server board. For example:

```
CP02
  $ContinueOnError
  ;       Install Mail Service
  $Run [Sys]<Sys>MailServer.run
  $FrontPanel (47)
  $EndBoard
```

In this Sysinit.jcl fragment, the Mail Service installs on the CP02 board, while the shared resource processor server board executes the **$FrontPanel** and **$ContinueOnError** keywords.

The following SysInit.jcl file includes a subset of possible board configurations. Furthermore, the example contains more boards than can fit in one cabinet; it is only an example rather than an actual working file. Available board memory should be taken into consideration when creating shared resource processor SysInit files. Real mode processor boards should have 768K memory to install services.

```
$Job      OsMaster
 $FrontPanel (30)
 ;        Displays 30 to front panel of SRP

 GP00
 $ContinueOnError
 ;        Install Queue Manager
 $Run [Sys]<Sys>InstallQMgr.Run,y,30
 $FrontPanel (31)
 ;        Install Byte Stream Service
 $Run [Sys]<Sys>InstallBatchBS.run
 $FrontPanel (32)
 ;        Install Batch Manager
          $Run [Sys]<Sys>InstallBatch.run,,BATCH
 $FrontPanel (33)
 $EndBoard

CP01
 $ContinueOnError
 ;        Install Font Service
 $Run [Sys]<Sys>FontService.run,[Sys]<Gps>ScreenFont.dbs,11264
 $FrontPanel (34)
 ;        Install GPS
 $Run [Sys]<Gps>GpsInstall.run
 $FrontPanel (35)
 $EndBoard
 FP00
 $ContinueOnError
 ;        Install BNet II
 $Run [Sys]<Sys>BNet.run,NodeX
 $FrontPanel (36)
 $Run [Sys]<Sys>NS.run
 $FrontPanel (37)
          ;Install Classic BNet
 $Run [Sys]<Sys>Net.run,NodeX,1
 $FrontPanel (38)
 $EndBoard
```

```
TP01
 $ContinueOnError
 ;        Install Classic BNet Async Ports
 $Run [Sys]<Sys>HDLC.run, 1
 $FrontPanel (39)
 $Run [Sys]<Sys>Async.run, 2
 $FrontPanel (40)
 $Run [Sys]<Sys>X25NISL.run, 3
 $FrontPanel (41)
 ;        Install Modem Server
 $Run [Sys]<Sys>ModemServer.run
 $FrontPanel (42)
 $EndBoard


TP00
 $ContinueOnError
 ;        Install OSI X25 Gateway
 $Run [Sys]<Sys>LAPB.run
 $FrontPanel (43)
 $Run [Sys]<Sys>X25PktInstall.run
 $FrontPanel (44)
 ;        Install OSI Transport WAN
 $Run [Sys]<Sys>Transport.run
 $FrontPanel (45)
 ;        Install OSI Session Service
 $Run [Sys]<Sys>Session.run
 $FrontPanel (46)
 $EndBoard

CP02
 $ContinueOnError
 ;        Install Mail Service
 $Run [Sys]<Sys>MailServer.run
 $FrontPanel (47)
 $EndBoard

FP01
 $ContinueOnError
 ;        Install QIC Tape Service
 $Run [Sys]<Sys>InstallQICService.run
 $FrontPanel (48)
 $EndBoard
```

```
DP00
 $ContinueOnError
 ;       Install 1/2" Tape Service
 $Run [sys]<sys>CreatePartition.run, 180k, Tape
 $FrontPanel (3A)
 $Run [sys]<sys>mInstallService.run, Tape, [sys]<sys>TapeService.run
 $FrontPanel (3B)
 $EndBoard

CP00
 $ContinueOnError
 ;       Install ClusterShare 1.3 Service
 $Run [sys]<sys>clustershare.run, 4, 4, 6, 6, 6, 2, 200, 2, 4, 2, N
 $FrontPanel (3C)
 $EndBoard

 $End
```

The following SysInit.jcl file segment uses the **RunNoWait** statement:

```
 .
 .
 .
 P01
 $ContinueOnError
 ;       Install Font Service
 $Run [Sys]<Sys>FontService.run, [Sys]<GPS>ScreenFont.dbs, 11264
 $FrontPanel (31)
 ;       Install Remote User Manager Service
 ;       RUM.run must be last run file on board
 $RunNoWait [Sys]<Sys>RUM.run
 .
 .
 .
```

The following SysInit.jcl file segment is an example using a SignOn.run file. Any workstation that boots will logon with the user name Camille, if the file [Sys]<Sys>Camille.user is available.

```
.
;       Logon as Camille
$Run [Sys]<Sys>SignOn.run, Camille
.
```

If you use this statement in a shared resource processor sysinit.x.jcl file, the user name is filled in at the Clusterview or Administrator Clusterview SignOn screen.

Glossary

A

access control.

The preventive enforcement of a security policy by the limiting of access to a system's resources.

application.

A collection of programs that perform a complete user function.

application partition.

A memory area within an operating system, reserved for executing an application.

application partition management.

An operating system facility that allows simultaneous execution of applications, each in its own secondary partition and provides for the creation, removal, loading, and management of secondary application partitions.

B

background.

An operating system mode for functions that can run unattended, without access to or control of the screen and keyboard.

background application partition.

A memory partition used by noninteractive applications, such as Batch Manager, which do not require access to or control of the keyboard and screen.

Batch control block.

A control block containing the batch job name and class, batch job control file handle and logical address, SysIn and SysOut bytestream work area, and buffers.

Batch job stream.

A file containing Job Control Language (JCL) statements, used by Batch Manager to process the noninteractive applications under its control.

Batch partition.

A background partition into which you install Batch Manager.

BNet.

A network communications software program that allows users to access the files and other resources of other systems at remote locations.

boot.

See Bootstrap

bootstrap.

The operation of starting a system by loading or reloading the operating system from a disk.

bytestream.

A readable (input) or writable (output) sequence of 8-bit bytes within the Sequential Access Method.

C

cluster.

See Cluster Configuration.

cluster configuration.

A local resource-sharing group of workstations containing a server (formerly master) and one or more cluster workstations.

cluster view.

Also known as a Remote Keyboard/Video Service (RKVS), allows one or more users from cluster workstations to execute run files at the server and remote nodes.

command.

An instruction to the operating system to perform a specific action.

command form.

An interactive display appearing after a command entry, requesting additional information with parameter prompts and entry fields.

configuration file.

A file specifying the characteristics of parallel printers, serial printers, or other devices connected to a workstation's communication ports.

D

default.

A predetermined value the system uses, or action the system takes, unless it receives an instruction to use an alternate value or perform an alternate action.

disk.

A reusable magnetic device for storing information; may be rigid (hard) or flexible (floppy). Amount of storage on a disk depends on its size.

disk drive.

A magnetic storage device that uses a hard or flexible disk to record information in the form of electromagnetic signals.

diskette.

A flexible (floppy) disk.

display.

A computer output screen that temporarily stores and presents graphic and/or textual data.

dollar sign ($)

The dollar sign ($) precedes an executable JCL command. It is optional when using Batch Manager.

dynamic queue.

An instruction from Batch Manager Queue Manager to create a partition and load Batch Manager into it. If Batch entries are absent from a Queue.Index file, dynamic queues are created.

DWord.

A double word.

E

editor.

An operating system application, activated by the **Edit** Command, that uses the keyboard to edit text appearing on a display; may be used to create text files containing JCL statements.

erc.

An abbreviation of error code. It is the most recent error code the system encountered. Each time a new ERC is encountered, it replaces the previous one.

exit run file.

A user-specified file that the system loads and activates when an application exits; each application partition has its own exit run file.

F

field.

An area on a command form for entering parameters or other units of information.

file.

A document, program, or other set of related data stored as a unit in a directory on a single volume.

file name.

A unique name describing the contents of a file. The name can contain 50 alphanumeric characters maximum, comprised of uppercase and lowercase letters, periods, hyphens, and right angle brackets (>).

file prefix.

Part of a file name that identifies a subdirectory, appears at the beginning of a file name, and is preceded by a right angle bracket (>).

file specification.

A complete file description of the form {Node}[VolumeName]<DirectoryName>FileName.

file suffix.

Part of a file name that further identifies the file, appears at the end of the file name, and consists of a period, hyphen, or right angle bracket (>), followed by three or more alphabetic characters.

foreground (or primary).

An operating system mode for functions that require access to and control of the screen and keyboard. All editing tasks are performed in foreground mode.

foreground partition.

An application partition which supports interactive programs using the keyboard and screen for execution.

function key.

A dedicated or variable key on the keyboard that initiates a specific, predetermined function.

I

initialization.

A process performed at the beginning of a program to ensure that all indicators and constants are set to prescribed conditions and values before the program is used.

input.

An instruction sent to the system.

install.

The action of copying software from installation media to the hard disk on a workstation.

Installation Manager.

The program that initiates the installation or deinstallation of software; these actions can be collecting information, verifying parameters, copying installation files to temporary locations, and setting up the video. All this initiation is then forwarded to Batch.

intrinsic.

An Executive command which has no run file associated with it (for example, **Copy**, its run file is defined as !2 rather than Copy.Run).

K

keyword.

The word in a JCL statement that identifies the statement. For example, a **Run** statement can contain several parameters and the keyword "Run."

L

linker.

An operating system utility that links one or more object modules into a run file.

log file.

A file that sequentially records Batch Manager processing activity and log statement entries for background execution.

long-lived memory.

A high-speed work area in a partition, used to pass parameters or data from one application to a succeeding application in the same partition.

M

merge.

An operation that allows the repeated combining of records within two or more files.

N

node.

The name of a workstation within a communication network. A node must have the communications software installed.

O

operating system.

A software program that provides the computer's basic instructions, including the loading and concurrent operation of individual programs, scheduling of system processes, and management of stored information.

P

parameter.

A variable or constant value transferred to and from a JCL statement, as required to execute a batch job.

parameter passing.

The operation of submitting parameters specified in a JCL statement to each job step.

partition.

A memory area within an operating system. May be either application or system.

process.

A batch job that is running.

Q

queue.

A special file containing entries for items waiting to be processed, such as batch jobs.

queue entry.

A formatted description of a job waiting to be processed.

queue index file.

A special file containing entries that define a system's queues.

Queue Manager.

An operating system utility that coordinates the processing sequence of the batch job entries within a queue.

R

Remote User Manager (RUM).

Allows Cluster View (the Remote Keyboard/Video Service - RKVS) to connect more than one application at the server to a workstation on the cluster. Multiple users or sessions can run applications on the same server board. RUM needs to be installed via a SysInit.jcl file at each board where multiple sessions are to be run. It must be the last system service installed on the board.

run file.

A memory image of a task in relocatable form, linked into the standard format; a complete program.

S

sequential access method.

An access method that emulates a conceptual, sequential character-oriented device known as a bytestream; provides device-independent access to devices.

server workstation.

The hub of a cluster configuration that provides a file system, queue management facility, and other services to other cluster workstations; supports its own interactive and batch applications.

service.

A program or subprogram that performs and manages tasks for other programs. Services have also been known as servers.

short-lived memory.

A work area in a partition used to contain code and data, such as input/output buffers and the heap.

spooler.

An operating system utility that manages the operations of printers assigned to its control; uses queues to store print requests until the printer becomes available.

stand-alone.

A type of workstation that uses its own hard disk for storage and system services.

status code.

A message display indicating the success or failure of a requested operation. See Appendix A, Troubleshooting, for a list of applicable messages.

submit.

An operating system facility that allows substituting a sequence of characters from a file for characters typed at the keyboard.

system administrator.

A person responsible for planning, generating, extending, and controlling operations to improve overall productivity.

system partition.

A memory partition containing the operating system and other dynamically installed services and facilities.

T

task.

A program consisting of executable code, data, and one or more processes.

task image.

A program stored in a run file, containing code segments and/or static data segments.

text file.

A file containing bytes that represent printable characters (such as letters, numbers, and punctuation) and/or control characters (such as tab and new line). A text file is the medium for creating JCL files using Editor or other word processing system.

U

user file.

A file on the system directory that identifies the user and specifies the system environment after the user signs on and exits from an application.

W

workstation.

The combination of a display screen, central processing unit (CPU), and keyboard (with or without local storage facilities).

Index

C

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W